Distributed Differentially Private Averaging with Improved Utility and Robustness to Malicious Parties

César Sabater, Aurélien Bellet, Jan Ramon

Distributed Differentially Private Averaging with Improved Utility and Robustness to Malicious Parties

César Sabater
Inria, France
first.last@inria.fr

Aurélien Bellet
Inria, France
first.last@inria.fr

Jan Ramon
Inria, France
first.last@inria.fr

Abstract

Learning from data owned by several parties, as in federated learning, raises challenges regarding the privacy guarantees provided to participants and the correctness of the computation in the presence of malicious parties. We tackle these challenges in the context of distributed averaging, an essential building block of distributed and federated learning. Our first contribution is a novel distributed differentially private protocol which naturally scales with the number of parties. The key idea underlying our protocol is to exchange correlated Gaussian noise along the edges of a network graph, complemented by independent noise added by each party. We analyze the differential privacy guarantees of our protocol and the impact of the graph topology, showing that we can match the accuracy of the trusted curator model even when each party communicates with only a logarithmic number of other parties chosen at random. This is in contrast with protocols in the local model of privacy (with lower accuracy) or based on secure aggregation (where all pairs of users need to exchange messages). Our second contribution is to enable users to prove the correctness of their computations without compromising the efficiency and privacy guarantees of the protocol. Our construction relies on standard cryptographic primitives like commitment schemes and zero knowledge proofs.

1 Introduction

Individuals are producing ever growing amounts of personal data, which in turn fuel innovative services based on machine learning (ML). The classic *centralized* paradigm consists in collecting, storing and analyzing this data on a (supposedly trusted) central server or in the cloud. which poses well documented privacy risks for the users. With the increase of public awareness and regulations, we are witnessing a shift towards a more *decentralized* paradigm where personal data remains on each user's device, see the recent trend of federated learning [50]. In this setting, users do not trust the central server (if any), or each other, which introduces new issues regarding privacy and security. First, the information shared by the participants during the decentralized training protocol can reveal a lot about their private data (see [59, 61] for inference attacks on federated learning). Formal guarantees such as differential privacy (DP) [31] are needed to provably mitigate this and convince privacy-aware users to participate in the protocol. Second, *malicious* users may send incorrect results to bias the learned model in arbitrary ways [45, 10, 3]. Robustness to such adversaries is crucial to persuade service providers to move to a more decentralized and privacy-friendly setting.

In this work, we tackle these challenges in the context of private distributed averaging. In this canonical problem, the objective is to privately compute an estimate of the average of values owned by many users who do not want to disclose them. Beyond simple data analytics, distributed averaging is of high relevance to modern ML. Indeed, it is the essential primitive used to aggregate user updates in gradient-based distributed and federated learning algorithms [56, 65, 58, 48, 2]. It also allows to train ML models whose sufficient statistics are averages (e.g., linear models and decision

Preprint. Under review.

trees). Distributed averaging with differential privacy guarantees has thus attracted a lot of interest in recent years. In the strong model of local differential privacy (LDP) [53, 30, 51, 52, 49], each user randomizes its input locally before sending it to an untrusted aggregator. Unfortunately, the best possible error for the estimated average with n users is a factor of $O(\sqrt{n})$ larger than in the centralized model of DP where a trusted curator aggregates data in the clear and perturbs the output [22]. To fill this gap, some work has explored relaxations of LDP that make it possible to match the accuracy of the trusted curator model. This is achieved through the use of cryptographic primitives such as secret sharing [32], secure aggregation [23, 64, 14, 48] and secure shuffling [34, 25, 6]. Many of these primitives however assume that all users truthfully follow the protocol (they are *honest-but-curious*), and they are generally intractable when the number of parties is large.

Our contribution is twofold. First, we propose a novel differentially private averaging protocol which can match the accuracy of the trusted curator setting while naturally scaling to a large number of users. Our approach, called GOPA (GOssip Noise for Private Averaging) by analogy with gossip protocols [16], is simple and decentralized. The main idea is to have users exchange correlated Gaussian noise along the edges of a network (represented as a connected graph) so as to mask their private values without affecting the global average. This (ultimately canceling) noise is complemented by the addition of independent (non-canceling) Gaussian noise by each user. We analyze the privacy of GOPA by modeling the knowledge of the adversary (colluding malicious users) as a system of linear equations and show that the privacy guarantees depend on the branching factor of a spanning tree of the subgraph of honest-but-curious users. Remarkably, we establish that we can recover the privacy-utility trade-off of the trusted curator setting as long as the graph of honest-but-curious users is connected and the pairwise-correlated noise variance is large enough. We further show that if the graph is well-connected, this variance can be significantly reduced. Finally, to ensure scalability and robustness in practice, we propose an efficient randomized procedure to construct a graph in which each user needs to communicate with only a *logarithmic* number of other users while still matching the privacy-utility trade-off of the trusted curator model. We prove these guarantees by leveraging and adapting results from random graph theory on embedding spanning trees in random graphs.

Our second contribution is a procedure to make GOPA verifiable by untrusted external parties, i.e., to provide users with the means of proving the correctness of their computations without compromising the efficiency or the privacy guarantees of the protocol. Our construction relies on commitment schemes and zero knowledge proofs (ZKPs), which are very popular in auditable electronic payment systems as well as cryptocurrencies. These cryptographic primitives, originally formalized in [43], scale well both in network communication and computational requirements and are perfectly suitable in our untrusted decentralized setting. We use classic and state-of-the-art ZKPs to design a procedure for the generation of noise with verifiable distribution, and ultimately to prove the integrity of the final computation (or detect malicious users who did not follow the protocol). Crucially, the privacy guarantees of the protocol are not compromised by this procedure (not even relaxed through a cryptographic hardness assumption), while the integrity of the computation relies on a standard discrete logarithm assumption. In the end, our protocol offers correctness guarantees that are essentially equivalent to the case where a service provider would itself hold the private data of users.

The paper is organized as follows. Section 2 introduces the problem setting as well as the adversary and privacy models. Section 3 presents the GOPA protocol, and we analyze its differential privacy guarantees in Section 4. We present our procedure to ensure correctness against malicious behavior in Section 5, and summarize computational and communication costs in Section 6. Finally, we discuss the related work in more details in Section 7, and conclude with future lines of research in Section 8.

2 Notations and Setting

We consider a set $U = \{1, \dots, n\}$ of $n \geq 3$ users (parties). Each user $u \in U$ holds a private value X_u, which can be thought of as being computed from the private dataset of user u. We assume that X_u lies in a bounded interval of $\mathbb{R}$ (without loss of generality, we assume $X_u \in [0, 1]$). The extension to the vector case is straightforward. We denote by X the column vector $X = [X_1, \dots, X_n]^\top \in [0, 1]^n$ of private values. Unless otherwise noted, all vectors are column vectors. The users communicate over a network represented by a connected undirected graph $G = (U, E)$, where $\{u, v\} \in E$ indicates that users u and v are neighbors in G and can exchange messages. For a given user u, we denote by $N(u) = \{v : \{u, v\} \in E\}$ the set of its neighbors.

The users aim to collaboratively compute the average value $X^{avg} = \frac{1}{n}\sum_{u=1}^{n} X_u$ without revealing their individual private value. Such a protocol can be readily used to privately execute distributed ML algorithms that interact with data through averages over values computed locally by the participants, but do not actually need to see the individual values. We give two concrete examples below.

Example 1 (Linear regression)**.** *Let $\lambda \geq 0$ be a public parameter. Each user u holds a private feature vector $\phi_u = [\phi_u^1, \ldots, \phi_u^d] \in \mathbb{R}^d$ and a private label $y_u \in \mathbb{R}$. The goal is to solve a ridge regression task, i.e. find $\theta^* \in \arg\min_\theta \frac{1}{n}\sum_{u\in U}(\phi_u^\top \theta - y_u)^2 + \lambda\|\theta\|^2$. The solution θ^* can be computed in closed form from the quantities $\frac{1}{n}\sum_{u\in U}\phi_u^i y_u$ and $\frac{1}{n}\sum_{u\in U}\phi_u^i\phi_u^j$ for all $i, j \in \{1, \ldots, d\}$.*

Example 2 (Federated ML)**.** *In federated learning [50] and more generally distributed empirical risk minimization, each user u holds a private dataset $\mathcal{D}_u$ and the goal is to find θ^* such that $\theta^* \in \arg\min_\theta \frac{1}{n}\sum_{u\in U} f(\theta; \mathcal{D}_u)$ where f is some loss function. Popular algorithms [56, 65, 58, 48, 2] all follow the same high-level procedure: at round t, each user u computes a local update θ_u^t based on $\mathcal{D}_u$ and the current global model θ^{t-1}, and the updated global model is computed as $\theta^t = \frac{1}{n}\sum_u \theta_u^t$.*

Threat model. We consider two commonly adopted adversary models, which were formalized by [42] and are used in the design of many secure protocols. An *honest-but-curious* (*honest* for short) user will follow the protocol specification, but may use all the information obtained during the execution to infer information about other users. In contrast, a *malicious user* may deviate from the protocol execution by sending incorrect values at any point (we assume that they follow the required communication policy; if not, this can be easily detected). Malicious users can collude, and thus will be seen as a single malicious party (the *adversary*) who has access to all information collected by malicious users (privacy with respect to a single honest user can be obtained as a special case). Our privacy guarantees will hold under the assumption that honest users communicate through secure channels, while the correctness of our protocol will be guaranteed under some form of the Discrete Logarithm Assumption (DLA), a standard assumption in cryptography (see Appendix B for details).

Each user in the network is either honest or malicious. Honest users do not know whether other nodes are malicious. We denote by $U^H \subseteq U$ the set of honest users, by $n_H = |U^H|$ the number of honest users and by $\rho = n_H/n$ their proportion in the network. We also denote by $G^H = (U^H, E^H)$ the subgraph of G induced by the set of honest users U^H, i.e., $E^H = \{\{u, v\} \in E : u, v \in U^H\}$. The properties of G and G^H will play a key role in the efficiency and privacy guarantees of our protocol.

Privacy model. Our goal is to design a protocol which satisfies differential privacy [31], which has become a gold standard notion in privacy-preserving information release.

Definition 1 (Differential privacy)**.** *Let $\varepsilon > 0, \delta \geq 0$. A (randomized) protocol $\mathcal{A}$ is (ε, δ)-differentially private if for all neighboring datasets X, X', i.e., datasets differing only in a single data point, and for all sets of possible outputs $\mathcal{O}$, we have:*

$$Pr(\mathcal{A}(X) \in \mathcal{O}) \leq e^{\varepsilon} Pr(\mathcal{A}(X') \in \mathcal{O}) + \delta. \tag{1}$$

3 GOPA: Gossip Noise for Private Averaging

In this section we describe our protocol, called GOPA (GOssip noise for Private Averaging). The high-level idea of GOPA is to have each user u mask its private value by adding two different types of noise. The first is a sum of pairwise-correlated noise terms $\Delta_{u,v}$ over the set of neighbors $v \in N(u)$ such that each $\Delta_{u,v}$ cancels out with the $\Delta_{v,u}$ of user v in the final result. The second type of noise is an independent term η_u which does not cancel out in the final result. At the end of the protocol, each user has generated a noisy version $\hat{X}_u$ of his private value X_u, which takes the following form:

$$\hat{X}_u = X_u + \textstyle\sum_{v\in N(u)} \Delta_{u,v} + \eta_u. \tag{2}$$

Algorithm 1 presents the detailed steps. Neighboring nodes $\{u, v\} \in E$ contact each other to draw a real number from the Gaussian distribution $\mathcal{N}(0, \sigma_\Delta^2)$, that u adds to its private value and v subtracts. Each user thereby distributes noise masking his private value across several users, which will provide some robustness to malicious parties. The idea is reminiscent of one-time pads in secure aggregation (see [14], Section 3) but we use Gaussian noise rather than padding and restrict exchanges to the edges of the graph instead of considering all pairs. As in gossip algorithms [16], the pairwise exchanges can be performed asynchronously and in parallel. Additionally, every user $u \in U$ adds an independent noise term $\eta_u \sim \mathcal{N}(0, \sigma_\eta^2)$ to its private value. This noise will ensure that the final estimate of the average satisfies differential privacy (see Section 4). σ_Δ^2 and σ_η^2 are public parameters of the protocol.

Algorithm 1 GOPA protocol

Input: graph $G = (U, E)$, private values $(X_u)_{u \in U}$, variances $\sigma_\Delta^2, \sigma_\eta^2 \in \mathbb{R}^+$

1: **for all** neighbor pairs $(u, v) \in E$ s.t. $u < v$ **do**
2: u and v draw a random $x \sim \mathcal{N}(0, \sigma_\Delta^2)$ and set $\Delta_{u,v} \leftarrow x$, $\Delta_{v,u} \leftarrow -x$
3: **for all** users $u \in U$ **do**
4: u draws a random $\eta_u \sim \mathcal{N}(0, \sigma_\eta^2)$ and reveals noisy value $\hat{X}_u \leftarrow X_u + \sum_{v \in N(u)} \Delta_{u,v} + \eta_u$

Utility of GOPA. The protocol generates a set of noisy values $\hat{X} = [\hat{X}_1, \ldots, \hat{X}_n]^\top$ which are then publicly released. They can be sent to an untrusted aggregator, or averaged in a decentralized way via gossiping [16]. In any case, the estimated average is given by $\hat{X}^{avg} = \frac{1}{n} \sum_{u \in U} \hat{X}_u = X^{avg} + \frac{1}{n} \sum_{u \in U} \eta_u$, which has expected value X^{avg} and variance σ_η^2/n. Ideally, we would like the total amount of independent noise to be of the same order as that needed to protect the average with the standard Gaussian mechanism in the trusted curator model of DP [33], that is $\eta_u = O(1/n)$.

Impact of σ_Δ^2. As we shall see in Section 4, the role of the pairwise noise terms is to compensate the independent noise that would be needed to protect individual values in the local model of differential privacy (where each user would release a locally perturbed input without communicating with other users). While the variance σ_Δ^2 must be sufficiently large to serve this purpose, it should be kept to a reasonable value. Indeed, a larger σ_Δ^2 induces a small (logarithmic) increase in communication costs as the representation space of the real values needs to be large enough to avoid overflows with high probability (we discuss these aspects in more details in further sections).

An important advantage of using centered Gaussian noise with bounded variance is to limit the impact of some users dropping out during the protocol. In particular, any residual noise term $\Delta_{u,v}$ (i.e., which does not cancel out due to either u or v dropping out) has expected value 0 and can be bounded with high probability. The smaller σ_Δ^2, the smaller the impact on the accuracy of the final output in the event of drop outs. If the accuracy is deemed insufficient, it is however possible to "roll back" some of the faulty noise exchanges to regain precision at the expense of extra communication.

4 Privacy Guarantees of GOPA

In this section, we show that GOPA achieves (ϵ, δ)-DP as long as G^H (the subgraph of honest users) is connected and $\sigma_\eta^2, \sigma_\Delta^2$ are large enough. More specifically, we will prove that (i) the variance σ_η^2 of the independent (non-canceling) noise can be as small as in the trusted curator setting, and (ii) the required variance σ_Δ^2 for the pairwise (canceling) noise depends on the topology of G^H.

4.1 Privacy Guarantees for Worst and Best Case Graphs

The knowledge acquired by the adversary (colluding malicious users) during the execution of the protocol consists of the following: (i) the noisy values $\hat{X}$ of all users, (ii) the full network graph G, (iii) the private value X_u and the noise η_u of the malicious users, and (iv) all $\Delta_{u,v}$'s for which u or v is malicious. The only unknowns are the private value X_u and independent noise η_u of each honest user $u \in U^H$, as well as the $\Delta_{u,v}$'s exchanged between honest users $\{u, v\} \in E^H$. From the knowledge above, the adversary can subtract the sum of noise exchanges $\sum_{v \in N(u) \setminus N^H(u)} \Delta_{u,v}$ from $\hat{X}_u$ to obtain $\hat{X}_u^H = X_u + \sum_{u \in N^H(u)} \Delta_{u,v} + \eta_u$ for every honest $u \in U^H$. The view of the adversary can be summarized by the vector $\hat{X}^H = (\hat{X}_u^H)_{u \in U^H}$ and correlation between its elements.

Our goal is to prove a differential privacy guarantee for GOPA. Adapting Definition 1 to our setting, for any input X and any possible outcome $\hat{X} \in \mathbb{R}^n$, we need to compare the probability of the outcome being equal to $\hat{X}$ when a honest user $v_1 \in U^H$ participates in the computation with private value $X_{v_1}^A$ to the probability of obtaining the same outcome when the value of v_1 is exchanged with an arbitrary value $X_{v_1}^B \in [0, 1]$. For notational simplicity, we denote by X^A the vector of private values $(X_u)_{u \in U^H}$ of honest users in which a user v_1 has value $X_{v_1}^A$, and by X^B the vector where v_1 has value $X_{v_1}^B$. X^A and X^B differ in only in the v_1-th coordinate, and their maximum difference is 1.

Our first result gives a differential privacy guarantee for the worst-case topology.

Theorem 1 (Privacy guarantee for the worst-case graph)**.** *Let X^A and X^B be two databases (i.e., graphs with private values at the vertices) which differ only in the value of one user. Let $\varepsilon, \delta \in (0,1)$ and $\theta = \frac{1}{\sigma_\eta^2 n_H} + \frac{n_H}{3\sigma_\Delta^2}$. If G^H is connected and the following two inequalities hold:*

$$\varepsilon \geq \theta/2 + \theta^{1/2}, \tag{3}$$

$$(\varepsilon - \theta/2)^2 \geq 2\log(2/\delta\sqrt{2\pi})\theta, \tag{4}$$

then GOPA *is (ε, δ)-differentially private, i.e., $P(\hat{X} \mid X^A) \leq e^\varepsilon P(\hat{X} \mid X^B) + \delta$.*

Crucially, Theorem 1 holds as soon as the subgraph G^H of honest users is connected.[1] In order to get a constant ε, inspecting the term θ shows that the variance σ_η^2 must be of order $1/n_H$ regardless of the topology of G^H. This is in fact optimal as it corresponds to the amount of noise required when averaging n_H values in the trusted curator setting. It also matches the amount of noise required when using secure aggregation with differential privacy in the presence of colluding users, where honest users need to add n/n_H more noise to compensate for collusion [64]. In order to match the privacy-utility trade-off of the trusted curator setting, further inspection of the equations in Theorem 1 shows that the variance σ_Δ^2 of the pairwise noise must be large enough. How large it must be depends in fact on the structure of the graph G^H. Theorem 1 describes the worst case, which is attained when every node has as few neighbors as possible while still being connected, i.e., when G^H is a path. In this case, Theorem 1 shows that the variance σ_Δ^2 needs to be of order n_H. Recall that this noise cancels out, so it does not impact the utility of the final output but only has a minor effect the communication cost and the robustness to drop out, as discussed in Section 3.

Intuitively, in graphs with higher connectivity, it should however be possible for σ_Δ^2 to be smaller than $O(n_H)$. Before considering more practical topologies, we illustrate this on the complete graph case, which provides a best-case scenario in terms of connectivity. Theorem 2 shows that for a fully connected G^H, σ_Δ^2 can be of order $1/n_H$, which is a *quadratic* reduction compared to the path case.

Theorem 2 (Privacy guarantee for the complete graph)**.** *Let $\varepsilon, \delta \in (0,1)$ and let G^H be the complete graph. If ε, δ, σ_η and σ_Δ satisfy the inequalities* (3) *and* (4) *in Theorem 1 with $\theta = \frac{1}{\sigma_\eta^2 n_H} + \frac{1}{\sigma_\Delta^2 n_H}$ then* GOPA *is (ε, δ)-differentially private.*

4.2 Privacy Guarantees for Practical Random Graphs

Our results so far gives two options that are not fully satisfactory from the practical perspective when the number of users n is large. One option is to come up with a graph G such that G^H is connected with high probability and use a large σ_Δ^2 of $O(n_H)$ as specified by the worst-case of Theorem 1. Alternatively, we can work with the complete graph, which allows smaller $O(1/n_H)$ variance (Theorem 2) but is intractable as all n^2 pairs of users need to exchange noise.

Instead, we propose a simple randomized procedure to construct a sparse network graph G such that G^H will be well-connected with high probability, and prove a DP guarantee for the whole process (random graph generation followed by GOPA) under much less noise than the worst-case. The idea is to make each (honest) user select k other users uniformly at random among all users. Then, the edge $\{u, v\} \in E$ is created if u selected v or v selected u (or both). This is known as a *random k-out graph* [13] or *random k-orientable graph* [36]. Such graphs are known for their very good connectivity properties [36, 67] and are used for instance in creating secure communication channels in distributed sensor networks [21]. We note that GOPA can be conveniently executed while constructing the random k-out graph. We have the following privacy guarantees.

Theorem 3 (Privacy guarantee for random k-out graphs)**.** *Let $\epsilon, \delta \in (0,1)$ and let G be obtained by letting all (honest) users randomly choose $k \leq n$ neighbors. Let k and $\rho = n_H/n$ be such that $\rho n \geq 81$, $\rho k \geq 4\log(2\rho n/3\delta)$, $\rho k \geq 6\log(\rho n/3)$ and $\rho k \geq \frac{3}{2} + \frac{9}{4}\log(2e/\delta)$. If ϵ, δ, σ_η and σ_Δ satisfy the inequalities* (3) *and* (4) *in Theorem 1 with*

$$\theta = n_H^{-1}\sigma_\eta^{-2} + \Big(\frac{1}{\lfloor (k-1)\rho/3 \rfloor - 1} + (12 + 6\log(n_H))/n_H\Big)\sigma_\Delta^{-2},$$

then GOPA *is $(\epsilon, 3\delta)$-differentially private.*

[1] Note that if G^H is not connected, we obtain a similar but weaker result for each connected component separately (n_H is replaced by the size of the connected component).

Complete		Random k-out			Worst-case	
$\rho = 1$	$\rho = 0.5$		$\rho = 1$	$\rho = 0.5$	$\rho = 1$	$\rho = 0.5$
1.7	2.1	Theorem 3:	44.7 ($k = 105$)	34.4 ($k = 203$)	9655.0	6114.8
		Simulation:	34.7 ($k = 20$)	28.4 ($k = 40$)		

Table 1: Value of σ_Δ to ensure (ϵ, δ)-DP with trusted curator utility for $n = 10000$, $\varepsilon = 0.1$, $\delta' = 1/n_H^2$, $\delta = 10\delta'$ depending on the topology, as obtained from Corollary 1.

This result has a similar form as Theorems 1-2 but requires k to be large enough (of order $\log(\rho n)/\rho$) so that G^H is sufficiently connected. Importantly, it is sufficient to have σ_Δ^2 of order $1/k\rho$ to ensure that we match the utility of the trusted curator model. Furthermore, each user needs to exchange with only $2k = O(\log n)$ other users in expectation, which is much more practical than a complete graph.

Scaling the noise in practice. Using the above results, we can precisely quantify the amount of independent and pairwise noise needed to achieve a desired privacy guarantee depending on the topology, as illustrated by the following corollary.

Corollary 1. *Let $\varepsilon, \delta' \in (0, 1)$, and $\sigma_\eta^2 = c^2/n_H\varepsilon^2$, where $c^2 > 2\log(1.25/\delta')$. Given some $\kappa > 0$, let $\sigma_\Delta^2 = \kappa\sigma_\eta^2$ if G is complete, $\sigma_\Delta^2 = \kappa\sigma_\eta^2 n_H(\frac{1}{\lfloor (k-1)\rho/3 \rfloor - 1} + (12 + 6\log(n_H))/n_H)$ if it is a random k-out graph, and $\sigma_\Delta^2 = \kappa\sigma_\eta^2 n_H^2/3$ for an arbitrary connected G^H. Then, GOPA is (ε, δ)-DP with $\delta \geq a(\delta'/1.25)^{\kappa/\kappa+1}$, where $a = 3.75$ for the random k-out graph and 1.25 otherwise.*

In Corollary 1, σ_η^2 is set such that after all noisy values are aggregated, the variance of the residual noise matches the one required by the standard Gaussian Mechanism [33] to achieve (ε, δ')-DP for an average of n_H values in the *centralized* setting. The privacy-utility trade-off achieved by GOPA is thus the same as in the trusted curator model up to a small constant in δ, as long as the pairwise variance σ_Δ^2 is large enough. As expected, we see that as $\sigma_\Delta^2 \to +\infty$ (that is, as $\kappa \to +\infty$), we have $\delta \to \delta'$. Given the desired $\delta \geq \delta'$, we can use Corollary 1 to determine a value for σ_Δ^2 that is sufficient for GOPA to achieve (ϵ, δ)-DP. Table 1 shows a numerical illustration with δ only a factor 10 larger than the δ' of the trusted curator setting. For random k-out graphs, we report the values of σ_Δ and k given by Theorem 3, as well as smaller (yet admissible) values obtained by numerical simulation (see Appendix A.5). We see that although the conditions of Theorem 3 are a bit conservative (we are confident that constants in our analysis can be improved), they still lead to practical values. Clearly, random k-out graphs provide a useful trade-off in terms of scalability and robustness.

Proof techniques. At the technical level, our analysis decomposes into a generic (ε, δ)-differential privacy result, where we model the knowledge of the adversary as a system of linear equations, and a topology-specific part. In the latter part, the generic result is instantiated by identifying a spanning tree in G^H with appropriate branching factor. While the optimal spanning tree is easy to construct for the worst-case and complete graphs, the case of random graphs is more involved and requires specific tools. We prove our result by leveraging and adapting results on embedding spanning trees in random graphs [55]. The full derivations and more detailed explanations can be found in Appendix A.

5 Ensuring Correctness Against Malicious Users

While the privacy guarantees of Section 4 hold no matter what the malicious users do, the utility guarantees of GOPA discussed in Section 3 are not valid if malicious users tamper with the protocol. In this section, we add to our protocol the capability of being audited to ensure the correctness of the computations while preserving privacy guarantees. While it is impossible to force a user to give the "right" input to the algorithm, this also holds in the centralized setting. Our goal is thus to guarantee that given the input vector X, GOPA will either generate a truthfully computed output $\hat{X}^{avg}$ or identify malicious behavior. Concretely, users will be able to prove the following properties:

$$\hat{X}_u = X_u + \Delta_u + \eta_u, \qquad \forall u \in U, \tag{5}$$

$$\Delta_{u,v} = -\Delta_{v,u}, \qquad \forall \{u, v\} \in E, \tag{6}$$

$$\eta_u \sim \mathcal{N}(0, \sigma_\eta^2), \qquad \forall u \in U, \tag{7}$$

$$X_u \in [0, 1], \qquad \forall u \in U. \tag{8}$$

We also explain how to verify consistency over multiple runs of GOPA on the same/related data.

Tools for verifying computations. Our approach consists in publishing an encrypted log of the computation using *cryptographic commitments* and proving that it is performed correctly without revealing any additional information using *zero knowledge proofs*. Commitments and ZKPs are popular in a number of applications such as privacy-friendly auditable financial systems [68, 38].

Commitments, first introduced in [12], allow users to commit to chosen values while keeping them hidden from others. After the commitment is performed, the committer cannot change its value, but can later reveal it or prove properties of it. For our protocol we use an optimized version [39] of the Pedersen commitment scheme [62]. A commitment is obtained by transforming the hidden statement using a hard-to-invert injective function Com, so that the recipient cannot know the original statement but can be sure that the committer cannot change the statement and still obtain the same commitment. Pedersen commitments additionally satisfy the *homomorphic property*, meaning that $Com(v_1 + v_2) = Com(v_1) + Com(v_2)$ for all values v_1, v_2. This property facilitates verifying the correctness of summations without revealing them: in our case, verifying Properties (5) and (6).

To verify other properties than the additive ones, we use another family of cryptographic operations that are known as Zero Knowledge Proofs (ZKPs). Informally speaking, ZKPs allow a user (prover) to effectively prove a true statement (completeness), but also allow the other user (verifier) to discover with high probability if a cheating prover is trying to prove a statement which is not true (soundness), in such a way that by performing the proof no information other than the proven statement is revealed (zero knowledge). Here, we use classic [24, 57] and more recent [18] proof techniques as building blocks. In particular, these building blocks support the verification of Properties (5) and (6), and constitute the bulk of the verification of (7) and (8).

Roughly speaking, the only assumption needed to ensure the validity of the commitments and ZKPs on which we rely is the Discrete Logarithm Assumption (DLA). We refer the interested reader to Appendices B.1– B.3, where we provide a step-by-step, bottom-up presentation of DLA, commitments, ZKPs, and of our own constructions over these existing building blocks.

Verification protocol. In a setup phase, users agree on a shared Pedersen commitment function Com. Then, while executing GOPA, users publish commitments of X_u and η_u for all $u \in U$ and of $\Delta_{u,v}$ for all $\{u, v\} \in E$. Due to the homomorphic property of the Pedersen commitments, everyone can then verify the summation relations, i.e., Properties (5) and (6). In a second phase, ZKPs are performed to prove the remaining parts of the verification. To limit the number of ZKPs while convincing all involved users of the correctness of the computations, publicly randomly generated challenges can be used. We implement the publication of commitments using a public bulletin board so that any party can verify the validity of the protocol, avoiding the need for a trusted verification entity. Users sign their messages so they cannot deny them. More general purpose distributed ledger technology such as in Bitcoin [60] could be used here, but we aim at an application-specific, light-weight and hence more scalable solution. Since the basic cryptographic operations are integer-based, we use a fixed precision scheme (see Appendix E for a discussion), which is efficient as our computations are mostly additive. More details on our verification protocol are provided in Appendix B.4.

It is easy to see that the correctness of the computation is guaranteed if Properties (5)-(8) are satisfied. Note that, as long as they are self-canceling and not excessively large (avoiding overflows and additional costs if a user drops out, see Appendix B.4), we do not need to ensure that pairwise noise terms $\Delta_{u,v}$ have been drawn from the prescribed distribution, as these terms do not influence the final result and only those involving honest users affect the privacy guarantees of Section 4. In contrast, Properties (7) and (8) are necessary to prevent a malicious user from biasing the outcome of the computation. Indeed, (7) ensures that the independent noise is generated correctly, while (8) ensures that input values are in the allowed range. The following result summarizes the security guarantees.

Theorem 4 (Security guarantees of GOPA)**.** *Under the DLA, a user $u \in U$ that passes the verification procedure proves that $\hat{X}_u$ was computed correctly. Additionally, u does not reveal any additional information about X_u by running the verification protocol, even if DLA does not hold.*

Consistency across multiple runs. Many ML algorithms (see Examples 1 and 2), require computing several averages involving the same or related private values. Additional security can be obtained by (a) requiring users to commit once to a value and then use it during the complete run of one or several algorithms, and (b) using state-of-the-art ZKP techniques [15, 18] to prove arithmetic relations.

We conclude by claiming that, with the above guarantees, we provide a similar control over the correctness of computations as in the centralized setting. We further discuss this in Appendix B.4.

6 Computation and Communication Costs

Our cost analysis, detailed in Appendix D, considers user-centered costs, which is natural as most operations can be performed asynchronously and in parallel. The following summarizes our results.

Theorem 5 (Complexity of GOPA)**.** *Let $B > 0$ be the desired precision of the fixed precision representation, e.g., the number 1 would be represented as the closest integer to $1/B$. Then, in the private averaging phase, every user u performs $O(|N(u)| + \log(1/B))$ computations and communications. In the verification phase, the protocol proves that all computations were performed correctly, where the η_u are drawn from a Gaussian distribution approximated with $1/B$ equiprobable bins and proved with precision B, with a cost of $O(|N(u)| + \log(1/B)\log(\log(1/B)))$ per user.*

Unlike other frameworks such as fully homomorphic encryption and secure multiparty computation, the cost of computing Pedersen commitments and ZKPs is tractable, even in massive amounts, as seen from production environment benchmarks [39, 18]. The above theorem assumes all users complete their participation in the protocol. If a user u drops out (gets off-line), we can roll back all computations where that user was involved (costing a broadcast and $O(1)$ per neighbor), or accept a potential bias or additional error in the result (corresponding to the $\Delta_{u,v}$ random variables).

7 Related Work

Local differential privacy (LDP) [53, 30, 51, 52, 49] require users to locally randomize their input before they send it to an untrusted aggregator. This very strong model of privacy comes at a significant cost in terms of the accuracy of the resulting estimates, see [70, 66] for the limits of machine learning in the local model. For the problem of averaging, the best possible error is of $O(1/n)$ in the trusted curator model while it is $O(1/\sqrt{n})$ in the local differential privacy model [22]. This makes the local model useful only in large-scale industrial settings where the number of participants is huge [35, 28].

Our approach belongs to the recent line of work which attempts to relax the LDP model so as to improve utility without relying on a trusted curator. Several papers have leveraged secret sharing and/or secure aggregation schemes to recover the utility of the trusted curator model, see [32, 64, 14, 23] for protocols and [48] for a concrete application to distributed machine learning. These approaches can be made robust to malicious parties, using for instance verifiable secret sharing [32]. However, they require $O(n)$ communication per party, which is not feasible beyond a few hundred or thousand participants. In contrast, our protocol requires only *logarithmic* communication. Recently, the shuffle model of privacy [25, 34, 6, 44, 5, 41], where inputs are passed to a secure shuffler that obfuscates the source of the messages, has been studied theoretically as an intermediate point between the local and trusted curator models. For differentially private averaging, the shuffle model allows to match the utility of the trusted curator setting. However, practical implementations of secure shuffling are not discussed in these works. Existing solutions typically rely on multiple layers of routing servers [29] with high communication overhead and non-collusion assumptions. Anonymous communication is also potentially at odds with the identification of malicious parties. To the best of our knowledge, all protocols for averaging in the shuffle model assume honest-but-curious parties. Finally, the recent work by [47] uses correlated Gaussian noise to achieve trusted curator accuracy for averaging. A key difference with our work is that the noise terms in their protocol are correlated at the global level (i.e., sums to zero once averaged over all users), requiring a call to a secure aggregation primitive that does not scale well, as discussed above. In contrast, our pairwise-canceling noise can be easily generated in a decentralized fashion. [47] also assume that all parties are honest-but-curious.

In summary, our protocol provides a unique combination of three important features: (a) accuracy of trusted curator setting, (b) logarithmic communication per user, and (c) robustness to malicious users.

8 Conclusion

We proposed GOPA, a protocol to privately compute averages over the values of many users. GOPA satisfies differential privacy, can match the utility of the trusted curator setting, and is robust to malicious parties. It can be readily used in distributed and federated ML [48, 50] as an alternative to more costly secure aggregation schemes. In future work, we would like to extend the scope of our approach beyond averaging, e.g. to robust aggregation rules for distributed SGD [11] and to U-statistics [8]. A promising direction is to combine GOPA with the functional mechanism [69].

Broader Impact

Our work promotes increased privacy and security in distributed and federated ML. The potential longer-term benefits of our work in this respect are a wider adoption of privacy-preserving ML solutions by service providers (thanks to the improved utility and the correctness guarantees), as well as better confidence of users and the general public in the ability of decentralized ML systems to avoid catastrophic data leaks. With this in mind, we plan to implement an efficient prototype to demonstrate the practicality of our approach.

Conversely, there are potential risks of accidental or deliberate misuse of our work in the sense that it could give a false sense of privacy to users if weak privacy parameters are used in deployment. This applies to all work on differential privacy. More applied research is needed towards developing a methodology to choose appropriate privacy parameters in a data-driven manner and to reliably assess the provided protection in practical use-cases. A good example of such work is a recent study by the US Census to prepare their adoption of differential privacy [1].

As ML penetrates into many areas of society, manipulations of algorithms by malicious parties could have (and have already had) serious societal and political consequences in the real world. A contribution of our work is to guarantee the correctness of the computation to prevent such manipulations. In contrast to the active and complementary research direction which aims to mitigate adversarial attacks with more robust algorithms, ensuring correctness by proving that the computation was done truthfully (without revealing the data) is not much studied in the ML community. This is likely because it requires technical tools from other fields (in particular cryptography and formal verification). We think our work, among others [46], could trigger more efforts in this direction.

Acknowledgments and Disclosure of Funding

The authors were supported by grants ANR-16-CE23-0016-01 and ANR-18-CE23-0018-03, by the European Union's Horizon 2020 Research and Innovation Program under Grant Agreement No.825081 COMPRISE and by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020. César Sabater is funded by an Inria Cordi-S scholarship.

The authors would like to thank James Bell, Pierre Dellenbach, Adrià Gascón and Alexandre Huat for fruitful discussions.

References

[1] John M. Abowd. The U.S. Census Bureau Adopts Differential Privacy. In *KDD*, 2018.

[2] Naman Agarwal, Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and Brendan McMahan. cpSGD: Communication-efficient and differentially-private distributed SGD. In *NeurIPS*, 2018.

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *AISTATS*, 2020.

[4] Victor Balcer and Salil Vadhan. Differential Privacy on Finite Computers. In *ITCS*, pages 43:1–43:21, 2018.

[5] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. Differentially Private Summation with Multi-Message Shuffling. Technical report, arxiv:1906.09116, 2019.

[6] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The Privacy Blanket of the Shuffle Model. In *CRYPTO*, 2019.

[7] Elaine B Barker and John Michael Kelsey. *Recommendation for random number generation using deterministic random bit generators (revised)*. NIST Special Publication (NIST SP) - 800-90A Rev 1, 2007.

[8] James Bell, Aurélien Bellet, Adrià Gascón, and Tejas Kulkarni. Private Protocols for U-Statistics in the Local Model and Beyond. In *AISTATS*, 2020.

[9] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge-based pseudo-random number generators. In *International Workshop on Cryptographic Hardware and Embedded Systems*, 2010.

[10] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin B. Calo. Analyzing federated learning through an adversarial lens. In *ICML*, 2019.

[11] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NIPS*, 2017.

[12] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, January 1983.

[13] Béla Bollobás. *Random Graphs (2nd edition)*. Cambridge University Press, 2001.

[14] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*, 2017.

[15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, Lecture Notes in Computer Science, pages 327–357, Berlin, Heidelberg, 2016. Springer.

[16] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE/ACM Transactions on Networking*, 14(SI):2508–2530, 2006.

[17] Stefan A Brands. *Rethinking public key infrastructures and digital certificates: building in privacy*. Mit Press, 2000.

[18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short Proofs for Confidential Transactions and More. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334, May 2018. ISSN: 2375-1207.

[19] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 93–118. Springer, 2001.

[20] Jan Camenisch and Markus Michels. Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, Lecture Notes in Computer Science, pages 107–122, Berlin, Heidelberg, 1999. Springer.

[21] Haowen Chan, Adrian Perrig, and Dawn Xiaodong Song. Random Key Predistribution Schemes for Sensor Networks. In *S&P*, 2003.

[22] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Optimal Lower Bound for Differentially Private Multi-party Aggregation. In *ESA*, 2012.

[23] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Privacy-preserving stream aggregation with fault tolerance. In *Financial Cryptography*, 2012.

[24] David Chaum and Torben Pryds Pedersen. Wallet Databases with Observers. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, Lecture Notes in Computer Science, pages 89–105, Berlin, Heidelberg, 1993. Springer.

[25] Albert Cheu, Adam D. Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed Differential Privacy via Shuffling. In *EUROCRYPT*, 2019.

[26] Sylvain Chevillard and Nathalie Revol. Computation of the error function erf in arbitrary precision with correct rounding. 2014. `https://hal.inria.fr/inria-00261360v2`.

[27] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In Yvo G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, Lecture Notes in Computer Science, pages 174–187, Berlin, Heidelberg, 1994. Springer.

[28] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *NIPS*, 2017.

[29] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[30] John C Duchi, Michael I Jordan, and Martin J Wainwright. Local privacy and statistical minimax rates. In *FOCS*, 2013.

[31] Cynthia Dwork. Differential Privacy. In *ICALP*, 2006.

[32] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *EUROCRYPT*, 2006.

[33] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[34] Ulfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, and Kunal Talwar. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *SODA*, 2019.

[35] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, 2014.

[36] Trevor I. Fenner and Alan M. Frieze. On the connectivity of random m-orientable graphs and digraphs. *Combinatorica*, 2(4):347–359, 1982.

[37] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, Lecture Notes in Computer Science, pages 186–194, Berlin, Heidelberg, 1987. Springer.

[38] Findora. `https://findora.org/`.

[39] Christian Franck and Johann Großschädl. Efficient Implementation of Pedersen Commitments Using Twisted Edwards Curves. In Samia Bouzefrane, Soumya Banerjee, Françoise Sailhan, Selma Boumerdassi, and Eric Renault, editors, *Mobile, Secure, and Programmable Networking*, Lecture Notes in Computer Science, pages 1–17, Cham, 2017. Springer International Publishing.

[40] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, Lecture Notes in Computer Science, pages 16–30, Berlin, Heidelberg, 1997. Springer.

[41] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Pure Differentially Private Summation from Anonymous Messages. Technical report, arXiv:2002.01919, 2020.

[42] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 1998.

[43] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, 18(1):186–208, February 1989. Publisher: Society for Industrial and Applied Mathematics.

[44] Valentin Hartmann and Robert West. Privacy-Preserving Distributed Learning with Secret Gradient Descent. Technical report, arXiv:1906.11993, 2019.

[45] Jamie Hayes and Olga Ohrimenko. Contamination attacks and mitigation in multi-party machine learning. In *NeurIPS*, 2018.

[46] Chris Hickey and Graham Cormode. Cheap Checking for Cloud Computing: Statistical Analysis via Annotated Data Streams. In *AISTATS*, 2018.

[47] Hafiz Imtiaz, Jafar Mohammadi, and Anand D. Sarwate. Distributed Differentially Private Computation of Functions with Correlated Noise. Technical report, arXiv:1904.10059, 2019.

[48] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Distributed learning without distress: Privacy-preserving empirical risk minimization. In *NeurIPS*, 2018.

[49] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. Discrete distribution estimation under local privacy. In *ICML*, 2016.

[50] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and Open Problems in Federated Learning. Technical report, arXiv:1912.04977, 2019.

[51] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Secure multi-party differential privacy. In *NIPS*, 2015.

[52] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal Mechanisms for Local Differential Privacy. *Journal of Machine Learning Research*, 17:1–51, 2016.

[53] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What Can We Learn Privately? In *FOCS*, 2008.

[54] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, November 2014. Google-Books-ID: OWZYBQAAQBAJ.

[55] Michael Krivelevich. Embedding spanning trees in random graphs. *SIAM J. Discret. Math.*, 24(4), 2010.

[56] Tao Lin, Sebastian U. Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't Use Large Mini-batches, Use Local SGD. In *ICLR*, 2020.

[57] Wenbo Mao. Guaranteed correct sharing of integer factorization with off-line shareholders. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, Lecture Notes in Computer Science, pages 60–71, Berlin, Heidelberg, 1998. Springer.

[58] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[59] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *S&P*, 2019.

[60] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online at http://bitcoin.org/bitcoin.pdf, 2008.

[61] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE Symposium on Security and Privacy*, 2019.

[62] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, 1991.

[63] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. On monotone formula closure of SZK. In *FOCS*, pages 454–465, 1994.

[64] Elaine Shi, T.-H. Hubert Chan, Eleanor G. Rieffel, Richard Chow, and Dawn Song. Privacy-Preserving Aggregation of Time-Series Data. In *NDSS*, 2011.

[65] Sebastian U. Stich. Local SGD Converges Fast and Communicates Little. In *ICLR*, 2019.

[66] Di Wang, Marco Gaboardi, and Jinhui Xu. Empirical Risk Minimization in Non-interactive Local Differential Privacy Revisited. In *NeurIPS*, 2018.

[67] Osman Yağan and Armand M. Makowski. On the Connectivity of Sensor Networks Under Random Pairwise Key Predistribution. *IEEE Transactions on Information Theory*, 59(9):5754–5762, 2013.

[68] Zcash. `https://z.cash/`.

[69] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. Functional mechanism: Regression analysis under differential privacy. In *VLDB*, 2012.

[70] Kai Zheng, Wenlong Mou, and Liwei Wang. Collect at Once, Use Effectively: Making Non-interactive Locally Private Learning Possible. In *ICML*, 2017.

SUPPLEMENTARY MATERIAL

Appendix A Proofs of Differential Privacy Guarantees

Recall that each user u has a private value X_u, and generates $\hat{X}_u$ by adding pairwise noise terms $\bar{\Delta}_u = \sum_{v \in N(u)} \Delta_{u,v}$ (with $\Delta_{u,v} + \Delta_{v,u} = 0$) as well as independent noise η_u. All random variables $\Delta_{u,v}$ (with $u < v$) and η_u are independent.

We have the system of linear equations

$$\hat{X} = X + \bar{\Delta} + \eta,$$

where $\bar{\Delta} = (\bar{\Delta}_u)_{u \in U}$ and $\eta = (\eta_u)_{u \in U}$. An adversary may be able to determine all X_u, $\Delta_{u,v}$ and η_u for every non-honest user u. Such an adversary can hence fill part of the variables in the system of linear equations. Let $X^H = (X_u)_{u \in U^H}$ be the vector of private values restricted to the honest users. Let $\bar{\Delta}_u^H = \sum_{v \in N^H(u)} \Delta_{u,v}$ and let $\hat{X}_u^H = \hat{X}_u - \sum_{v \in N(u) \setminus N^H(u)} \Delta_{u,v}$. In this way, the adversary should now solve a new system of linear equations:

$$\hat{X}^H = X^H + \bar{\Delta}^H + \eta^H,$$

where $\bar{\Delta}^H = (\bar{\Delta}_u^H)_{u \in U^H}$ and $\eta^H = (\eta_u)_{u \in U^H}$.

The expectation and covariance matrix of $\hat{X}^H$ are respectively given by:

$$\mathbb{E}\left[\hat{X}^H\right] = X^H \text{ and } \Sigma = \Sigma_{\hat{X}^H} = \sigma_\eta^2 I_{U^H} + \sigma_\Delta^2 \Lambda - \sigma_\Delta^2 A = \sigma_\eta^2 I_{U^H} + \sigma_\Delta^2 L,$$

where $I_{U^H} \in \mathbb{R}^{n_H \times n_H}$ is the identity matrix, $A \in \mathbb{R}^{n_H \times n_H}$ is the adjacency matrix of G^H (the communication graph restricted to honest users), Λ is a diagonal matrix with $\Lambda_{u,u} = \sum_{v \in U^H} A_{u,v}$, and L is the graph Laplacian matrix of G^H.

Now consider the real vector space Z of dimension $n_H + |E^H|$ of all possible values of independent and pairwise noise terms of honest users. For convenience, we will write $(\eta^H, \Delta^H) \in Z$ to denote that the independent noise terms take values $\eta^H = (\eta_u)_{u \in U^H}$ and the pairwise noise terms take values $\Delta^H = (\Delta_{u,v})_{\{u,v\} \in E^H}$. Let $K \in \mathbb{R}^{n_H \times |E^H|}$ denote the oriented incidence matrix of the graph G^H such that $KK^\top = L$ and for any pairwise noise values $\Delta^H \in \mathbb{R}^{|E^H|}$, $K\Delta^H = \bar{\Delta}^H$. For the sake of readability, in the following we drop the superscript H for $(\eta^H, \Delta^H) \in Z$ as it will clear from the context that we work in the space Z.

Let

$$\Sigma^{(g)} = \begin{bmatrix} \sigma_\eta^2 I_{U^H} & 0 \\ 0 & \sigma_\Delta^2 I_{E^H} \end{bmatrix}.$$

We then have a joint probability distribution of independent Gaussians:

$$P((\eta, \Delta)) = C_1 \exp\left(-\frac{1}{2}(\eta, \Delta)^\top (\Sigma^{(g)})^{-1}(\eta, \Delta)\right),$$

where $C_1 = (2\pi)^{-(n_H + |E^H|)/2} |\Sigma^{(g)}|^{-1/2}$.

Consider the following subspaces of Z:

$$\begin{aligned} Z^A &= \{(\eta, \Delta) \in Z \mid \eta + K\Delta = \hat{X}^H - X^A\}, \\ Z^B &= \{(\eta, \Delta) \in Z \mid \eta + K\Delta = \hat{X}^H - X^B\}. \end{aligned}$$

Assume that the (only) vertex for which X^A and X^B differ is v_1. We can assume without loss of generality that private values are in the interval $[0, 1]$, hence $X_{v_1}^A - X_{v_1}^B = 1$. Now choose any $t = (t_\eta, t_\Delta) \in Z$ such that $t_\eta + Kt_\Delta = X^A - X^B$. It follows that $Z^A = Z^B + t$, i.e., $Y \in Z^A$ if and only if $Y + t \in Z^B$.

We split the rest of our derivations into four parts. In Section A.1, we first prove a generic intermediate result (Lemma 1) which gives differential privacy guarantees for GOPA that depend on the particular

choice of t. Then, in Section A.2, we study the best and worst-case topologies and show how the choice of t can be made intelligently depending on the graph structure, leading to Theorem 1 and Theorem 2. Section A.3.4 presents the derivations for random k-out graphs leading to Theorem 3. Finally, Section A.4 shows how these results can be instantiated to match the accuracy of the centralized Gaussian mechanism (Corollary 1).

A.1 Generic Differential Privacy Result

Let $t = (t_\eta, t_\Delta) \in Z$ such that $t_\eta + Kt_\Delta = X^A - X^B$ as defined above. We start by proving differential privacy guarantees which depend on the particular choice of t.

Lemma 1. *Let $\varepsilon, \delta \in (0,1)$. Under the setting introduced above, if*

$$\varepsilon \geq (t^\top \Sigma^{(-g)} t)^{1/2} + t^\top \Sigma^{(-g)} t/2, \tag{9}$$

and

$$\frac{\left(\varepsilon - t^\top \Sigma^{(-g)} t/2\right)^2}{t^\top \Sigma^{(-g)} t} \geq 2 \log\left(\frac{2}{\delta\sqrt{2\pi}}\right), \tag{10}$$

where $\Sigma^{(-g)} = \left(\Sigma^{(g)}\right)^{-1}$, then GOPA *is (ε, δ)-differentially private, i.e.,*

$$P(\hat{X} \mid X^A) \leq e^\varepsilon P(\hat{X} \mid X^B) + \delta.$$

Proof. It is sufficient to prove that

$$\left|\log \frac{P((\eta, \Delta))}{P((\eta, \Delta) + t)}\right| \leq \varepsilon \tag{11}$$

with probability $1 - \delta$ over (η, Δ). Denoting $\gamma = (\eta, \Delta)$ for convenience, we need to prove that with probability $1 - \delta$ it holds that $|\log(P(\gamma)/P(\gamma + t))| \geq e^\varepsilon$. We have

$$\begin{aligned}
\left|\log \frac{P(\gamma)}{P(\gamma + t)}\right| &= \left| -\frac{1}{2}\gamma^\top \Sigma^{(-g)} \gamma + \frac{1}{2}(\gamma + t)^\top \Sigma^{(-g)} (\gamma + t)\right| \\
&= \left|\frac{1}{2}(2\gamma + t)^\top \Sigma^{(-g)} t\right|.
\end{aligned}$$

To ensure Equation (11) hold with probability $1 - \delta$, since we are interested in the absolute value, we will show that

$$P\Big(\frac{1}{2}(2\gamma + t)^\top \Sigma^{(-g)} t \geq \varepsilon\Big) \leq \delta/2,$$

which is equivalent to

$$P(\gamma \Sigma^{(-g)} t \geq \varepsilon - t^\top \Sigma^{(-g)} t/2) \leq \delta/2. \tag{12}$$

The variance of $\gamma \Sigma^{(-g)} t$ is

$$\begin{aligned}
\text{var}(\gamma \Sigma^{(-g)} t) &= \sum_v \text{var}\left(\eta_v \sigma_\eta^{-2} t_v\right) + \sum_e \text{var}\left(\Delta_e \sigma_\Delta^{-2} t_e\right) \\
&= \sum_v \text{var}\left(\eta_v\right) \sigma_\eta^{-4} t_v^2 + \sum_e \text{var}\left(\Delta_e\right) \sigma_\Delta^{-4} t_e^2 \\
&= \sum_v \sigma_\eta^2 \sigma_\eta^{-4} t_v^2 + \sum_e \sigma_\Delta^2 \sigma_\Delta^{-4} t_e^2 \\
&= \sum_v \sigma_\eta^{-2} t_v^2 + \sum_e \sigma_\Delta^{-2} t_e^2 \\
&= t^\top \Sigma^{(-g)} t.
\end{aligned}$$

For any centered Gaussian random variable Y with variance σ_Y^2, we have that

$$P(Y \geq \lambda) \leq \frac{\sigma_Y}{\lambda\sqrt{2\pi}} \exp\left(-\lambda^2 / 2\sigma_Y^2\right). \tag{13}$$

Let $Y = \gamma \Sigma^{(-g)} t$, $\sigma_Y^2 = t^\top \Sigma^{(-g)} t$ and $\lambda = \varepsilon - t^\top \Sigma^{(-g)} t/2$ so satisfying

$$\frac{\sigma_Y}{\lambda\sqrt{2\pi}} \exp\left(-\lambda^2 / 2\sigma_Y^2\right) \leq \delta/2 \tag{14}$$

implies (12). Equation (14) is equivalent to

$$\frac{\lambda}{\sigma_Y} \exp\left(\lambda^2/2\sigma_Y^2\right) \geq 2/\delta\sqrt{2\pi},$$

and taking logarithms we need

$$\log\left(\frac{\lambda}{\sigma_Y}\right) + \frac{1}{2}\left(\frac{\lambda}{\sigma_Y}\right)^2 \geq \log\left(\frac{2}{\delta\sqrt{2\pi}}\right).$$

To make this inequality hold, we require

$$\log\left(\frac{\lambda}{\sigma_Y}\right) \geq 0 \tag{15}$$

and

$$\frac{1}{2}\left(\frac{\lambda}{\sigma_Y}\right)^2 \geq \log\left(\frac{2}{\delta\sqrt{2\pi}}\right). \tag{16}$$

Equation (15) is equivalent to

$$\lambda \geq \sigma_Y.$$

Substituting λ and σ_Y we get

$$\varepsilon - t^\top \Sigma^{(-g)} t/2 \geq (t^\top \Sigma^{(-g)} t)^{1/2},$$

which is equivalent to (9). Substituting λ and σ_Y in Equation (16) gives (10). □

Essentially, given some ε, Equation (9) provides a lower bound for the noise (the diagonal of $\Sigma^{(g)}$) to be added. Equation (9) also implies that the lefthandside of Equation (10) is larger than 1. Equation (10) may then require the noise or ε to be even higher if $2\log(2/\delta\sqrt{2\pi}) \geq 1$, i.e., $\delta \leq 0.48394$.

A.2 Worst and Best Case Topologies

Lemma 1 holds for all vectors $t = (t_\eta, t_\Delta)$. There exist multiple vectors t such that $t_\eta + Kt_\Delta = X^A - X^B$, so we will select the vector t which allows us most easily to prove differential privacy using the above.

Let us first consider the worst case, which corresponds to Theorem 1.

Proof of Theorem 1. Let T be a spanning tree of the (connected) communication graph G^H. Let E^T be the set of edges in T. Let $t \in \mathbb{R}^{n_H + |E^H|}$ be a vector such that:

- For vertices $u \in U^H$, $t_u = 1/n_H$.
- For edges $e \in E^H \setminus E^T$, $t_e = 0$.
- Finally, for edges $e \in E^T$, we choose t_e in the unique way such that $t_\eta + Kt_\Delta = (X^A - X^B)$.

In this way, $t_\eta + Kt_\Delta$ is a vector with a 1 on the v_1 position and 0 everywhere else. We can find a unique vector t using this procedure for any communication graph.

It holds that

$$t_\eta^\top t_\eta = n_H \left(\frac{1}{n_H}\right)^2 = \frac{1}{n_H}. \tag{17}$$

In both Equations (9) and (10) of Lemma 1, a higher $t^\top \Sigma^{(-g)} t$ is 'worse' in the sense that a higher ε or δ is needed. Conversely, for a given ε and δ, $t^\top \Sigma^{(-g)} t$ must be sufficiently low to satisfy both

inequalities. We can see $t_\Delta^\top \sigma_\Delta^{-2} t_\Delta$ is maximized (thus producing the worst case) if the spanning tree T is a path $(v_1\, v_2 \ldots v_{n_H})$, in which case $t_{\{v_i,v_{i+1}\}} = (n_H - i)/n_H$. Therefore,

$$\begin{aligned} t_\Delta^\top t_\Delta &\leq \sum_{i=1}^{n_H-1} \left(\frac{n_H - i}{n_H}\right)^2 \\ &= \frac{n_H(n_H-1)(2n_H-1)/6}{n_H^2} \\ &= \frac{(n_H-1)(2n_H-1)}{6n_H}. \end{aligned} \tag{18}$$

We now plug in the specific values of t in our general results of Lemma 1. Consider a given δ, σ_η and σ_Δ. Substituting Equation (17) and (18) into Equation (9) yields

$$\varepsilon \geq \frac{1}{2}\left(\sigma_\eta^{-2}\frac{1}{n_H} + \sigma_\Delta^{-2}\frac{(n_H-1)(2n_H-1)}{6n_H}\right) + \left(\sigma_\eta^{-2}\frac{1}{n_H} + \sigma_\Delta^{-2}\frac{(n_H-1)(2n_H-1)}{6n_H}\right)^{1/2},$$

which is satisfied if

$$\varepsilon \geq \frac{1}{2\sigma_\eta^2 n_H} + \frac{n_H}{6\sigma_\Delta^2} + \left(\frac{1}{\sigma_\eta^2 n_H} + \frac{n_H}{3\sigma_\Delta^2}\right)^{1/2}. \tag{19}$$

Substituting Equation (17) and (18) into Equation (10) yields

$$\frac{\left(\varepsilon - \left(\sigma_\eta^{-2}\frac{1}{n_H} + \sigma_\Delta^{-2}\frac{(n_H-1)(2n_H-1)}{6n_H}\right)/2\right)^2}{\sigma_\eta^{-2}\frac{1}{n_H} + \sigma_\Delta^{-2}\frac{(n_H-1)(2n_H-1)}{6n_H}} \geq 2\log\left(\frac{2}{\delta\sqrt{2\pi}}\right),$$

which is satisfied if

$$\left(\varepsilon - \frac{1}{2\sigma_\eta^2 n_H} - \frac{n_H}{6\sigma_\Delta^2}\right)^2 \geq 2\log\left(\frac{2}{\delta\sqrt{2\pi}}\right)\left(\frac{1}{\sigma_\eta^2 n_H} + \frac{n_H}{3\sigma_\Delta^2}\right). \qquad \square \tag{20}$$

We can observe that in the worst case σ_Δ^2 should be large (linear in n_H) to keep ε small, which has no direct negative effect on the accuracy of the resulting $\hat{X}$. On the other hand, σ_η^2 can be small (of the order $1/n_H$), which means that independent of the number of participants or the way they communicate a small amount of independent noise is sufficient to achieve differential privacy.

However, the necessary value of σ_Δ^2 depends strongly on the network structure. This becomes clear in Theorem 2, which covers the case of the complete graph.

Proof of Theorem 2. If the communication graph is fully connected, we can use the following values for the vector t:

- As earlier, for $v \in U^H$, let $t_v = -1/n_H$.
- For edges $\{u, v\}$ with $v_1 \notin \{u, v\}$, let $t_{\{u,v\}} = 0$.
- For $u \in U^H \setminus \{v_1\}$, let $t_{\{u,v_1\}} = 1/n_H$.

Again, one can verify that $t_\eta + K t_\Delta = X^A - X^B$ is a vector with a 1 on the v_1 position and 0 everywhere else. In this way, again $t_\eta^\top t_\eta = 1/n_H$ but now $t_\Delta^\top t_\Delta = (n_H - 1)/n_H^2$ is much smaller. With this network structure, substituting into Equation (9) yields

$$\varepsilon \geq \frac{1}{2}\left(\sigma_\eta^{-2}\frac{1}{n_H} + \sigma_\Delta^{-2}\frac{(n_H-1)}{n_H^2}\right) + \left(\sigma_\eta^{-2}\frac{1}{n_H} + \sigma_\Delta^{-2}\frac{n_H-1}{n_H^2}\right)^{1/2},$$

which is satisfied if

$$\varepsilon \geq \frac{\sigma_\eta^{-2} + \sigma_\Delta^{-2}}{2n_H} + \left(\frac{\sigma_\eta^{-2} + \sigma_\Delta^{-2}}{n_H}\right)^{1/2}. \tag{21}$$

Substituting into Equation (10) yields

$$\frac{\left(\varepsilon - \left(\sigma_\eta^{-2}\frac{1}{n_H} + \sigma_\Delta^{-2}\frac{n_H-1}{n_H^2}\right)/2\right)^2}{\sigma_\eta^{-2}\frac{1}{n_H} + \sigma_\Delta^{-2}\frac{n_H-1}{n_H^2}} \geq 2\log\left(\frac{2}{\delta\sqrt{2\pi}}\right),$$

which is satisfied if

$$\left(\varepsilon - \left(\sigma_\eta^{-2} + \sigma_\Delta^{-2}\right)/2n_H\right)^2 \geq 2\log\left(\frac{2}{\delta\sqrt{2\pi}}\right)\frac{\sigma_\eta^{-2} + \sigma_\Delta^{-2}}{n_H}. \qquad \square \quad (22)$$

A practical communication graph will be between the two extremes of the path and the complete graph, as shown by the case of random k-out graphs (see Appendix A.3).

A.3 Random k-out Graphs

In this section, we will study the differential privacy properties for the case where all users select k neighbors randomly, leading to a proof of Theorem 3. We will start by analyzing the properties of G^H (Section A.3.1). Section A.3.2 consists of preparations for embedding a suitable spanning tree in G^H. Next, in Section A.3.3 we will prove a number of lemmas showing that such suitable spanning tree can be embedded almost surely in G^H. Finally, we will apply these results to proving differential privacy guarantees for GOPA when communicating over such a random k-out graph G in Section A.3.4, proving Theorem 3.

In this section, all newly introduced notations and definitions are local and will not be used elsewhere. At the same time, to follow more closely existing conventions in random graph theory, we may reuse in this section some variable names used elsewhere and give them a different meaning.

A.3.1 The Random Graph G^H

Recall that the communication graph G^H is generated as follows:

- We start with $n = |U|$ vertices where U is the set of agents.
- All (honest) agents randomly select k neighbors to obtain a k-out graph G.
- We consider the subgraph G^H induced by the set U^H of honest users. Recall that $n_H = |U^H|$ and that a fraction ρ of the users is honest, hence $n_H = \rho n$.

Let $k_H = \rho k$. The graph G^H is a subsample of a k-out-graph, which for larger n_H and k_H follows a distribution very close to that of Erdős-Rényi random graphs $G_p(n_H, 2k_H/n_H)$. To simplify our argument, in the sequel we will assume G^H is such random graph as this does not affect the obtained result. In fact, the random k-out model concentrates the degree of vertices more narrowly around the expected value then Erdős-Rényi random graphs, so any tail bound our proofs will rely on that holds for Erdős-Rényi random graphs also holds for the graph G^H we are considering. In particular, for $v \in U^H$, the degree of v is a random variable which we will approximate for sufficiently large n_H and k_H by a binomial $B(n_H, 2k_H/n_H)$ with expected value $2k_H$ and variance $2k_H(1 - 2k_H/n_H) \approx 2k_H$.

A.3.2 The Shape of the Spanning Tree

Remember that our general strategy to prove differential privacy results is to find a spanning tree in G^H and then to compute the norm of the vector t_Δ that will "spread" the difference between X^A and X^B over all vertices (so as to get a σ_η of the same order as in the trusted curator setting). Here, we will first define the shape of a rooted tree and then prove that with high probability this tree is isomorphic to a spanning tree of G^H. Of course, we make a crude approximation here, as in the (unlikely) case that our predefined tree cannot be embedded in G^H it is still possible that other trees could be embedded in G^H and would yield similarly good differentially privacy guarantees. While our bound on the risk that our privacy guarantee does not hold will not be tight, we will focus on proving our result for reasonably-sized U and k, and on obtaining interesting bounds on the norm of t_Δ.

Let $G^H = ([n_H], E^H)$ be a random graph where between every pair of vertices there is an edge with probability $2k_H/n_H$. The average degree of G^H is $2k_H$.

Let $k_H \geq 4$. Let $q \geq 3$ be an integer. Let $\Delta_1, \Delta_2 \dots \Delta_q$ be a sequence of positive integers such that

$$\left(\sum_{i=1}^{q}\prod_{j=1}^{i}\Delta_j\right) - (\Delta_q + 1)\prod_{j=1}^{q-2}\Delta_j < n_H \leq \sum_{i=1}^{q}\prod_{j=1}^{i}\Delta_j. \tag{23}$$

Let $T = ([n_H], E_T)$ be a balanced rooted tree with n_H vertices, constructed as follows. First, we define for each level l a variable z_l representing the number of vertices at that level, and a variable Z_l representing the total number of vertices in that and previous levels. In particular: at the root $Z_{-1} = 0$, $Z_0 = z_0 = 1$ and for $l \in [q-2]$ by induction $z_l = z_{l-1}\Delta_l$ and $Z_l = Z_{l-1} + z_l$. Then, $z_{q-1} = \lceil (n_H - Z_{q-2})/(\Delta_q + 1) \rceil$, $Z_{q-1} = Z_{q-2} + z_{q-1}$, $z_q = n_H - Z_{q-1}$ and $Z_q = n_H$. Next, we define the set of edges of T:

$$E_T = \{\{Z_{l-2} + i, Z_{l-1} + z_{l-1}j + i\} \mid l \in [q] \wedge i \in [z_{l-1}] \wedge z_{l-1}j + i \in [z_l]\}.$$

So the tree consists of three parts: in the first $q-2$ levels, every vertex has a fixed, level-dependent number of children, the last level is organized such that a maximum of parents has Δ_q children, and in level $q-1$ parent nodes have in general $\Delta_{q-1} - 1$ or Δ_{q-1} children. Moreover, for $0 \leq l \leq q-2$, the difference between the number of vertices in the subtrees rooted by two vertices in level l is at most $\Delta_q + 2$. We also define the set of children of a vertex, i.e., for $l \in [q]$ and $i \in [z_{i-1}]$,

$$ch(Z_{l-2} + i) = \{Z_{l-1} + z_{l-1}j + i \mid z_{l-1}j + i \in [z_l]\}.$$

In Section A.3.3, we will show conditions on n_H, Δ, k_H and δ such that for a random graph G^H on n_H vertices and a vertex v_1 of G^H, with high probability (at least $1 - \delta$) G^H contains a subgraph isomorphic to T whose root is at v_1.

A.3.3 Random Graphs Almost Surely Embed a Balanced Spanning Tree

The results below are inspired by [55]. We specialize this result to our specific problem, obtaining proofs which are also valid for graphs smaller than 10^{10} vertices, even if the bounds get slightly weaker when we drop terms of order $O(\log(\log(n_H)))$ for the simplicity of our derivation.

Let F be the subgraph of T induced by all its non-leafs, i.e., $F = ([\lceil n_H/\Delta \rceil], E_F)$ with $E_F = \{\{i,j\} \in E_T \mid i,j \leq Z_{q-1}\}$.

Lemma 2. *Let G^H and F be defined as above. Let v_1 be a vertex of G^H. Let $n_H \geq k_H \geq \Delta \geq 3$. Let $\gamma = \max_{l=1}^{q-1}\Delta_l/k_H$ and let $\gamma + 4(\Delta+2)^{-1} + 2n_H^{-1} \leq 1$. Let $k_H \geq 4\log(2n_H/\delta_F(\Delta_q+2))$. Then, with probability at least $1 - \delta_F$, there is an isomorphism ϕ from F to a subgraph of G^H, mapping the root 1 of F on v_1.*

Proof. We will construct ϕ by selecting images for the children of vertices of F in increasing order, i.e., we first select $\phi(1) = v_1$, then map children $\{2 \dots \Delta_1 + 1\}$ of 1 to vertices adjacent to v_1, then map all children of 2 to vertices adjacent to $\phi(2)$, etc. Suppose we are processing level $l \in [q-1]$ and have selected $\phi(j)$ for all $j \in ch(i')$ for all $i' < i$ for some $Z_{l-1} < i \leq Z_l$. We now need to select images for the Δ_l children $j \in ch(i)$ of vertex i (or in case $l = q-1$ possibly only $\Delta_l - 1$ children). This means we need to find Δ_l neighbors of i not already assigned as image to another vertex (i.e., not belonging to $\cup_{0 \leq i' < i}\phi(ch(i'))$). We compute the probability that this fails. For any vertex $j \in [n_H]$ with $i \neq j$, the probability that there is an edge between i and j in G^H is $2k_H/n_H$. Therefore, the probability that we fail to find Δ_l free neighbors of i can be upper bounded as

$$Pr\left[\textsc{Fail}_F(i)\right] = Pr\left[\text{Bin}\left(n_H - Z_l, \frac{2k_H}{n_H}\right) < \Delta_l\right] \leq \exp\left(\frac{-\left((n_H - Z_l)\frac{2k_H}{n_H} - \Delta_l\right)^2}{2(n_H - Z_l)\frac{2k_H}{n_H}}\right). \tag{24}$$

We know that $n_H - Z_l \geq z_q$. Moreover, $(z_q + \Delta_q - 1)/\Delta_q \geq z_{q-1}$ and $Z_{q-2} + 1 \leq z_{q-1}$, hence $2(z_q + \Delta_q - 1)/\Delta_q \geq Z_{q-2} + Z_{q-1} + 1 = Z_{q-1} + 1$ and $2(z_q + \Delta_q - 1)/\Delta_q + z_q \geq n_H + 1$. There follows

$$z_q(2 + \Delta_q) \geq n_H + 1 - 2(\Delta_q - 1)/\Delta_q \geq n_H - 1.$$

Therefore,

$$n_H - Z_l \geq z_q \geq n_H(1 - 2(\Delta_q + 2)^{-1} - n_H^{-1}). \tag{25}$$

Substituting this and $\Delta_l \geq \gamma k_H$ in Equation (24), we get

$$\begin{aligned} Pr\left[\textsc{Fail}_F(i)\right] &\leq \exp\left(\frac{-\left(n_H(1-2(\Delta_q+2)^{-1}-n_H^{-1})\frac{2k_H}{n_H} - k_H\gamma\right)^2}{2n_H(1-2(\Delta_q+2)^{-1}-n_H^{-1})\frac{2k_H}{n_H}}\right) \\ &\leq \exp\left(\frac{-k_H^2\left(2(1-2(\Delta_q+2)^{-1}-n_H^{-1})-\gamma\right)^2}{4k_H}\right) \\ &\leq \exp\left(\frac{-k_H^2}{4k_H}\right) = \exp\left(\frac{-k_H}{4}\right), \end{aligned}$$

where the latter inequality holds as $\gamma + 4(\Delta+2)^{-1} + 2n_H^{-1} \leq 1$. As $k_H \geq 4\log(2n_H/\delta_F(\Delta_q+2))$ we can conclude that

$$Pr\left[\textsc{Fail}_F(i)\right] \leq \frac{\delta(\Delta_q+2)}{2n_H}.$$

The total probability of failure to embed F in G^H is therefore given by

$$\begin{aligned} \sum_{i=2}^{Z_{q-1}} \textsc{Fail}_F(i) &\leq (Z_{q-1}-1)\frac{\delta_F(\Delta_q+2)}{2n_H} \\ &\leq \left(n_H(2(\Delta_q+2)^{-1}+n_H^{-1})-1\right)\frac{\delta_F(\Delta_q+2)}{2n_H} \\ &= \frac{2n_H}{\Delta_q+2}\frac{\delta_F(\Delta_q+2)}{2n_H} = \delta_F, \end{aligned}$$

where we again applied (25). □

Now that we can embed F in G^H, we still need to embed the leaves of T. Before doing so, we review a result on matchings in random graphs. The next lemma mostly follows [13, Theorem 7.11], we mainly adapt to our notation, introduce a confidence parameter and make a few less crude approximations.[2]

Lemma 3. *Let $m \geq 27$ (in our construction, $m = z_q$) and $\zeta \geq 4$. Consider a random bipartite graph with vertex partitions $A = \{a_1 \dots a_m\}$ and $B = \{b_1 \dots b_m\}$, where for every $i, j \in [m]$ the probability of an edge $\{a_i, b_j\}$ is $p = \zeta(\log(m))/m$. Then, the probability of a complete matching between A and B is higher then*

$$1 - \frac{em^{-2(\zeta-1)/3}}{1 - m^{-(\zeta-1)/3}}.$$

Proof. For a set X of vertices, let $\Gamma(X)$ be the set of all vertices adjacent to at least one member of X. Then, if there is no complete matching between A and B, there is some set X, with either $X \subset A$ or $X \subset B$, which violates Hall's condition, i.e., $|\Gamma(X)| < |X|$. Let X be the smallest set satisfying this property (so the subgraph induced by $X \cup \Gamma(X)$ is connected). The probability that such sets X and $\Gamma(X)$ of respective sizes i and $j = |\Gamma(X)|$ exist is upper bounded by appropriately combining:

- the number of choices for X, i.e., 2 (for selecting A or B) times $\binom{m}{i}$,
- the number of choices for $\Gamma(X)$, i.e., $\binom{m}{i-1}$,

[2] In particular, even though Bollobas's proof is asymptotically tight, its last line uses the fact that $(e\log n)^{3a}n^{1-a+a^2/n} = o(1)$ for all $a \leq n/2$. This expression is only lower than 1 for $n \geq 5.6 \cdot 10^{10}$, and as the sum of this expression over all possible values of a needs to be smaller than δ_F, we do not expect this proof applies to graphs representing current real-life datasets.

- an upper bound for the probability that under these choices of X and $\Gamma(X)$ there are at least $2i-2$ edges (as the subgraph induced by $X \cup \Gamma(X)$ is connected), i.e., $\left(\begin{array}{c} ij \\ i+j-1 \end{array} \right)$ possible choices of the vertex pairs and p^{i+j-1} the probability that these vertex pairs all form edges, and
- the probability that there is no edge between any of $X \cup \Gamma(X)$ and the other vertices, i.e., $(1-p)^{i(m-j)+j(m-i)} = (1-p)^{m(i+j)-2ij}$.

Thus, we upper bound the probability of observing such sets X and $\Gamma(X)$ of sizes i and j as follows:

$$\begin{aligned} \textsc{Fail}_B(i,j) &\leq \left(\begin{array}{c} m \\ i \end{array} \right) \left(\begin{array}{c} m \\ j \end{array} \right) \left(\begin{array}{c} ij \\ i+j-1 \end{array} \right) p^{i+j-1}(1-p)^{m(i+j)-2ij} \\ &\leq \left(\frac{me}{i}\right)^i \left(\frac{me}{j}\right)^j \left(\frac{ije}{i+j-1}\right)^{i+j-1} p^{i+j-1}(1-p)^{m(i+j)-2ij}. \end{aligned}$$

Here, in the second line the classic upper bound for combinations is used: $\left(\begin{array}{c} m \\ i \end{array} \right) < \left(\frac{me}{i}\right)^i$. As $2j \leq i+j-1$, we get

$$\begin{aligned} \textsc{Fail}_B(i,j) &\leq \left(\frac{me}{i}\right)^i \left(\frac{me}{j}\right)^j \left(\frac{ie}{2}\right)^{i+j-1} p^{i+j-1}(1-p)^{m(i+j)-2ij} \\ &\leq \frac{m^{i+j}e^{2i+2j-1}i^{j-1}}{j^j 2^{i+j-1}} p^{i+j-1}(1-p)^{m(i+j)-2ij}. \end{aligned} \tag{26}$$

As $0 < p < 1$, there also holds

$$(1-p)^{1/p} < 1/e,$$

and therefore

$$(1-p)^{m(i+j)-2ij} = (1-p)^{\frac{1}{p}p(m(i+j)-2ij)} < (1/e)^{p(m(i+j)-2ij)}.$$

We can substitute $p = \zeta(\log(m))/m$ to obtain

$$\begin{aligned} (1-p)^{m(i+j)-2ij} &< (1/e)^{(\zeta(\log(m))/m)(m(i+j)-2ij)} \\ &= \left(\frac{1}{m}\right)^{\zeta(m(i+j)-2ij)/m}. \end{aligned}$$

Substituting this and $p < \zeta \log(m)/m$ into Equation (26), we get

$$\begin{aligned} \textsc{Fail}_B(i,j) &\leq \frac{m^{i+j}e^{2i+2j-1}i^{j-1}}{j^j 2^{i+j-1}} \left(\frac{\log(m)\zeta}{m}\right)^{i+j-1} \left(\frac{1}{m}\right)^{\zeta(m(i+j)-2ij)/m} \\ &= \frac{mei^{j-1}}{j^j} \left(\frac{\log(m)\zeta e^2}{2}\right)^{i+j-1} \left(\frac{1}{m}\right)^{\zeta((i+j)-2ij/m)}. \end{aligned}$$

As $j < i < m/2$, $2 \leq i < (i+j) - 2ij/m < i+j$, there follows

$$\textsc{Fail}_B(i,j) \leq \frac{mei^{j-1}}{j^j} \left(\frac{\log(m)\zeta e^2}{2}\right)^{i+j-1} \left(\frac{1}{m}\right)^{\zeta((i+j)-2ij/m)}.$$

Given that $m^{\zeta/3} \geq \zeta \log(m) e^2/2$ holds for $m \geq 27$ and $\zeta \geq 4$, we get

$$\begin{aligned} \textsc{Fail}_B(i,j) &\leq \frac{mei^{j-1}}{j^j} \left(\frac{\log(m)\zeta e^2}{2m^{1/3}}\right)^{i+j-1} m^{-\zeta((i+j)-2ij/m)+\zeta(i+j-1)/3} \\ &\leq \frac{ei^{j-1}}{j^j} m^{-\zeta(\frac{2}{3}(i+j)+1/3-2ij/m)+1} \\ &\leq \frac{ei^{j-1}}{j^j} m^{-\zeta(\frac{1}{3}i+\frac{1}{3})+1}. \end{aligned}$$

As $\zeta \geq 4$, this implies

$$\textsc{Fail}_B(i,j) \quad \leq \quad \frac{e}{i}\left(\frac{i}{j}\right)^j m^{-\frac{\zeta i}{3}-\frac{1}{3}} \tag{27}$$

There holds:

$$\begin{aligned}
\sum_{j=1}^{i-1}\left(\frac{i}{j}\right)^j &= \sum_{j=1}^{\lfloor i/3\rfloor}\left(\frac{i}{j}\right)^j + \sum_{j=\lfloor i/3\rfloor+1}^{i}\left(\frac{i}{j}\right)^j \\
&\leq \sum_{j=1}^{\lfloor i/3\rfloor}\left(\frac{i}{j}\right)^j + \sum_{j=\lfloor i/3\rfloor+1}^{i} 3^j = \sum_{j=1}^{\lfloor i/3\rfloor}\left(\frac{i}{j}\right)^j + 3^{\lfloor i/3\rfloor+1}\frac{3^{i-\lfloor i/3\rfloor}-1}{3-1} \\
&< \sum_{j=1}^{\lfloor i/3\rfloor}\left(\frac{i}{j}\right)^j + \frac{3^{i+1}}{2} < \sum_{j=1}^{\lfloor i/3\rfloor} i^{i/3} + \frac{3^{i+1}}{2} \\
&\leq \frac{i}{3}i^{i/3} + \frac{3^{i+1}}{2}.
\end{aligned}$$

Substituting in Equation (27) gives

$$\begin{aligned}
\textsc{Fail}_B &= \sum_{i=2}^{m/2}\sum_{j=1}^{i-1}\textsc{Fail}_B(i,j) \\
&< \sum_{i=2}^{m/2}\sum_{j=1}^{i-1}\frac{e}{i}\left(\frac{i}{j}\right)^j m^{-\frac{\zeta i}{3}-\frac{1}{3}} < \sum_{i=2}^{m/2}\left(\frac{i^{i/3+1}}{3}+\frac{3^{i+1}}{2}\right)\frac{e}{i}m^{-\frac{\zeta i}{3}-\frac{1}{3}} \\
&< \sum_{i=2}^{m/2}\left(i^{i/3}+3^{i+1}\right)\frac{e}{2}m^{-\frac{\zeta i}{3}-\frac{1}{3}} < \sum_{i=2}^{m/2} i^{i/3}m^{-\frac{\zeta i}{3}-\frac{1}{3}} + 3^{i+1}\frac{e}{2}m^{-\frac{\zeta i}{3}-\frac{1}{3}}.
\end{aligned}$$

As $m \geq 27 = 3^3$ we can now write

$$\begin{aligned}
\textsc{Fail}_B &< \frac{e}{2}\sum_{i=2}^{m/2} m^{-\frac{(\zeta-1)i}{3}-\frac{1}{3}} + m^{(i+1)/3}m^{-\frac{\zeta i}{3}-\frac{1}{3}} < \frac{e}{2}\sum_{i=2}^{m/2} m^{-\frac{(\zeta-1)i}{3}-\frac{1}{3}} + m^{-\frac{(\zeta-1)i}{3}} \\
&< e\sum_{i=2}^{m/2} m^{-\frac{(\zeta-1)i}{3}} = em^{-\frac{2(\zeta-1)}{3}}\sum_{i=0}^{m/2-2}\left(m^{-\frac{\zeta-1}{3}}\right)^i = em^{-\frac{2(\zeta-1)}{3}}\frac{1-\left(m^{-\frac{\zeta-1}{3}}\right)^{m/2-1}}{1-\left(m^{-\frac{\zeta-1}{3}}\right)} \\
&< em^{-\frac{2(\zeta-1)}{3}}\frac{1}{1-\left(m^{-\frac{\zeta-1}{3}}\right)}.
\end{aligned}$$

This concludes the proof. □

Lemma 4. *Let $m \geq 27$ and $\delta_B > 0$. Let*

$$\zeta = \max\left(4, 1+\frac{3\log(2e/\delta_B)}{2\log(m)}\right).$$

Consider a random bipartite graph as described in Lemma 3 above. Then, with probability at least $1-\delta_B$ there is a complete matching between A and B.

Proof. From the given ζ, we can infer that

$$\begin{aligned}
\zeta - 1 &\geq \tfrac{3\log(2e/\delta_B)}{2\log(m)} \\
\zeta - 1 &\geq \tfrac{-3\log(\delta_B/2e)}{2\log(m)} \\
-\tfrac{2}{3}(\zeta-1)\log(m) &\leq \log\left(\delta_B/2e\right) \\
m^{-2(\zeta-1)/3} &\leq \delta_B/2e
\end{aligned}$$

We also know that $\zeta \geq 4$ and $m \geq 27$, hence $(\zeta-1)/3 \geq 1$ and $1 - m^{-(\zeta-1)/3} \geq 26/27 \geq 1/2$. We know from Lemma 3 that the probability of having a complete matching is at least

$$1 - \frac{em^{-2(\zeta-1)/3}}{1 - m^{-(\zeta-1)/3}} \geq 1 - \frac{e(\delta_B/2e)}{1/2} = 1 - \delta_B. \qquad \square$$

Lemma 5. *Let $\delta_B > 0$, $\Delta \geq 1$ (in our construction, $\Delta = \Delta_q$) and $m \geq 27$. Let $d_1 \dots d_l$ be positive numbers, with $d_i = \Delta$ for $i \in [l-1]$, $d_l \in [\Delta]$ and $\sum_{i=1}^{l} d_i = m$. Let $A = \{a_1 \dots a_l\}$ and $B = \{b_1 \dots b_m\}$ be disjoint sets of vertices in a random graph G^H where the probability to have an edge $\{a_i, b_j\}$ is $2k_H/n_H$ for any i and j. Let*

$$p \geq 1 - \left(1 - \max\left(4, 1 + \frac{3\log(2e/\delta_B)}{2\log(m)}\right)\frac{\log(m)}{m}\right)^{\Delta}. \tag{28}$$

Then with probability at least $1 - \delta_B$, G^H contains a collection of disjoint d_i-stars with centers a_i and leaves in B.

Proof. Define an auxiliary random bipartite graph G' with sides $A' = \{a'_1 \dots a'_m$ and $B = \{b_1 \dots b_m$. For every $i, j \in [m]$, the probability of having an edge between a_i and b_j in G' is $p' = 1-(1-p)^{1/\Delta}$. We relate the distributions on the edges of G^H and G' by requiring there is an edge between a_i and b_j if and only if there is an edge between $a'_{\Delta(i-1)+i'}$ and b_j for all $i' \in [\Delta]$.

From Equation (28) we can derive

$$p' \geq \max\left(4, 1 + \frac{3\log(2e/\delta_B)}{2\log(m)}\right)\frac{\log(m)}{m}. \tag{29}$$

Setting

$$\zeta = \max\left(4, 1 + \frac{3\log(2e/\delta_B)}{2\log(m)}\right),$$

this ensures p' satisfies the constraints of Lemma 4:

$$p' = \zeta\log(m)/m.$$

As a result, there is a complete matching in G' with probability at least $1 - \delta_B$, and hence the required stars can be found in G^H with probability at least $1 - \delta_B$. $\square$

Lemma 6. *Let $\delta_F > 0$ and $\delta_B > 0$. Let G^H and T and their associated variables be as defined above. Assume that the following conditions are satisfied:*

(a) $n_H \geq 27(\Delta_q + 2)/\Delta_q$,
(b) $\gamma + 2(\Delta_q + 2)^{-1} + n_H^{-1} \leq 1$,
(c) $k_H \geq 4\log(2n_H/\delta_F(\Delta_q + 2))$,
(d) $k_H \geq \max\left(4, 1 + \dfrac{3\log(2e/\delta_B)}{2\log(n_H\Delta_q/(\Delta_q+2))}\right)\dfrac{\Delta_q+2}{2}\log\left(\dfrac{n_H\Delta_q}{\Delta_q+2}\right)$,
(e) $\gamma = \max_{l=1}^{q-1}\Delta_l/k_H$.

Let G^H be a random graph where there is an edge between any two vertices with probability p. Let v_1 be a vertex of G^H. Then, with probability at least $1 - \delta_F - \delta_B$, there is a subgraph isomorphism between the tree T defined above and G^H such that the root of T is mapped on v_1.

Proof. The conditions of Lemma 2 are clearly satisfied, so with probability $1 - \delta_F$ there is a tree isomorphic to F in G^H. Then, from condition (d) above and knowing that the edge probability is $p = 2k_H/n_H$, we obtain

$$p \geq \max\left(4, 1 + \frac{3\log(2e/\delta_B)}{2\log(n_H\Delta_q/(\Delta_q+2))}\right)\frac{1}{n_H\Delta_q/(\Delta_q+2)}\log\left(\frac{n_H\Delta_q}{\Delta_q+2}\right)\Delta_q.$$

Taking into account that $m = n_H\Delta_q/(\Delta_q+2)$, we get

$$p \geq \max\left(4, 1+\frac{3\log(2e/\delta_B)}{2\log(m)}\right)\frac{1}{m}\log(m)\,\Delta_q,$$

which implies the condition on p in Lemma 4. The other conditions of that lemma can be easily verified. As a result, with probability at least $1-\delta_B$ there is a set of stars in G^H linking the leaves of F to the leaves of T, so we can embed T completely in G^H. □

A.3.4 Running GOPA on Random Graphs

Assume we run GOPA on a random graph satisfying the properties above, what can we say about the differential privacy guarantees? According to Lemma 1, it is sufficient that there exists a spanning tree and vectors t_η and t_Δ such that $t_\eta + Kt_\Delta = X^A - X^B$. We fix t_η in the same way as for the other discussed topologies (see Section A.2) in order to achieve the desired σ_η and focus our attention on t_Δ. According to Lemma 6, with high probability there exists in G^H a spanning tree rooted at the vertex where X^A and X^B differ and a branching factor Δ_l specified per level. So given a random graph on n_H vertices with edge density $2k_H/n_H$, if the conditions of Lemma 6 are satisfied we can find such a tree isomorphic to T in the communication graph between honest users G^H. In many cases (reasonably large n_H and k_H), this means that the lemma guarantees a spanning tree with branching factor as high as $O(k_H)$, even though it may be desirable to select a small value for the branching factor of the last level in order to more easily satisfy condition (d) of Lemma 6, e.g., $\Delta_q = 2$ or even $\Delta_q = 1$.

Lemma 7. *Under the conditions described above,*

$$\begin{aligned} t_\Delta^\top t_\Delta &\leq \frac{1}{\Delta_1}\left(1+\frac{1}{\Delta_2}\left(1+\frac{|}{\Delta_3}\left(\ldots\frac{1}{\Delta_q}\right)\right)\right)+\frac{(\Delta_q+2)(\Delta_q+2+2q)}{n_H} \qquad (30)\\ &\leq \frac{1}{\Delta_1}\left(1+\frac{2}{\Delta_2}\right)+O(n_H^{-1}). \end{aligned}$$

Proof. Let q be the depth of the tree T. The tree is balanced, so in every node the number of vertices in the subtrees of its children differs at most Δ_q+2. For edges e incident with the root (a level 0 node), $|t_e-\Delta_1^{-1}| \leq n_H^{-1}(\Delta_q+2)$. In general, for a node at level l (except leaves or parents of leaves), there are $\prod_{i=1}^{l}\Delta_i$ vertices, each of which have Δ_{l+1} children, and for every edge e connecting such a node with a child,

$$\left|t_e-\prod_{i=1}^{l+1}\Delta_i^{-1}\right| \leq (\Delta_q+2)/n_H.$$

For a complete tree (of $1+\Delta+\ldots+\Delta^q$ vertices), we would have

$$t_\Delta^\top t_\Delta = \sum_{l=1}^{q}\prod_{i=1}^{l}\Delta_i\left(\prod_{i=1}^{l}\Delta_i^{-1}\right)^2 = \sum_{l=1}^{q}\left(\prod_{i=1}^{l}\Delta_i\right)^{-1},$$

which corresponds to the first term in Equation (30). As the tree may not be complete, i.e., there may be less than $\prod_{i=1}^{q}\Delta_i$ leaves, we analyze how much off the above estimate is. For an edge e connecting a vertex of level l with one of its children,

$$\left|t_e-\prod_{i=1}^{l+1}\Delta_i\right| \leq (\Delta_q+2)/n_H,$$

and hence

$$\begin{aligned}
\left| t_e^2 - \left(\prod_{i=1}^{l+1} \Delta_i\right)^2 \right| &\leq \left(t_e^2 - t_e\left(\prod_{i=1}^{l+1} \Delta_i\right)\right) + \left(t_e\left(\prod_{i=1}^{l+1} \Delta_i\right) - \left(\prod_{i=1}^{l+1} \Delta_i\right)^2\right) \\
&\leq t_e(\Delta_q+2)/n_H + \left(\prod_{i=1}^{l+1} \Delta_i\right)(\Delta_q+2)/n_H \\
&\leq (\left(\prod_{i=1}^{l+1} \Delta_i\right) + n_H^{-1})(\Delta_q+2)n_H + \left(\prod_{i=1}^{l+1} \Delta_i\right)(\Delta_q+2)/n_H \\
&= (\Delta_q+2)^2/n_H^2 + 2\left(\prod_{i=1}^{l+1} \Delta_i\right)(\Delta_q+2)/n_H.
\end{aligned}$$

Summing over all edges gives

$$\begin{aligned}
t_\Delta^\top t_\Delta - \sum_{l=1}^{q}\left(\prod_{i=1}^{l} \Delta_i\right)^{-1} &\leq \sum_{l=1}^{q} z_l \left((\Delta_q+2)/n_H^2 + 2\left(\prod_{i=1}^{l+1} \Delta_i\right)^{-1}(\Delta_q+2)/n_H\right) \\
&= (\Delta_q+2)^2/n_H + \sum_{l=1}^{q} 2z_l\left(\prod_{i=1}^{l+1} \Delta_i\right)^{-1}(\Delta_q+2)/n_H \\
&\leq (\Delta_q+2)^2/n_H + \sum_{l=1}^{q} 2(\Delta_q+2)/n_H \\
&= \frac{(\Delta_q+2)(\Delta_q+2+2q)}{n_H}. \qquad \square
\end{aligned}$$

So if we choose parameters Δ for the tree T, the above lemmas provide a value δ such that T can be embedded in G^H with probability at least $1-\delta$ and an upper bound for $t_\Delta^\top t_\Delta$ that can be obtained with the resulting spanning tree in G^H.

Theorem 3 in the main text summarizes these results, simplifying the conditions by assuming that $\Delta_i = \lfloor (k-1)\rho/2 \rfloor$ for $i \leq q-1$ and $\Delta_q = 2$.

Proof of Theorem 3. Let us choose $\Delta_i = \lfloor (k-1)\rho/3 \rfloor$ for $i \in [q-1]$ and $\Delta_q = 1$ for some appropriate q such that Equation (23) is satisfied. We also set $\delta = \delta_F = \delta_B$.

Then, the conditions of Lemma 6 are satisfied. In particular, condition (a) holds as $n_H = \rho n \geq 81 = 27(\Delta_q+2)/\Delta_q$. Condition (e) implies that

$$\gamma = \max_{i=1}^{q-1} \Delta_i/k_H = \frac{1}{k_H}\left\lfloor \frac{(k-1)\rho}{3} \right\rfloor.$$

Condition (b) holds as

$$\begin{aligned}
\gamma + 2(\Delta_q+2)^{-1} + n_H^{-1} &= \frac{1}{k_H}\left\lfloor \frac{(k-1)\rho}{3} \right\rfloor + \frac{2}{3} + n_H^{-1} \\
&\leq \frac{1}{k_H}\frac{(k-1)\rho}{3} + \frac{2}{3} + n_H^{-1} \\
&\leq \frac{1}{3} - \frac{\rho}{3k_H} + \frac{2}{3} + n_H^{-1} = \frac{1}{3} - \frac{1}{3k} + \frac{2}{3} + n_H^{-1} \\
&\leq \frac{1}{3} - \frac{1}{n_H} + \frac{2}{3} + n_H^{-1} = 1.
\end{aligned}$$

Condition (d) holds because we know that $\rho k \geq 6\log(\rho n/3)$, which is equivalent to

$$k_H \geq 4\frac{\Delta+2}{\Delta}\log\left(\frac{n_H\Delta_q}{\Delta_q+2}\right),$$

and we know that $\rho k \geq \frac{3}{2} + \frac{9}{4}\log(2e/\delta)$, which is equivalent to

$$k_H \geq \left(1 + \frac{3\log(2e/\delta_B)}{2\log(n_H\Delta_q/(\Delta_q+2))}\right)\frac{\Delta+2}{\Delta}\log\left(\frac{n_H\Delta_q}{\Delta_q+2}\right).$$

Finally, condition (c) is satisfied as we know that $\rho k \geq 4\log(\rho n/3\delta)$. Therefore, applying the lemma, we can with probability at least $1-2\delta$ find a spanning tree isomorphic to T. If we find one, Lemma 7 implies that

$$\begin{aligned}
t_\Delta^\top t_\Delta &\leq \sum_{l=1}^{q}\left(\prod_{i=1}^{l}\Delta_i\right)^{-1} + \frac{(\Delta_q+2)(\Delta_q+2+2q)}{n_H} \\
&= \sum_{l=1}^{q-1}\Delta_1^{-l} + \Delta_1^{1-q}\Delta_q^{-1} + \frac{3(3+2q)}{n_H} = \Delta_1^{-1}\frac{1-\Delta_1^{1-q}}{1-\Delta_1^{-1}} + \Delta_1^{1-q}\Delta_q^{-1} + \frac{9+6q}{n_H} \\
&\leq \frac{1}{\Delta_1-1} + \frac{3}{n_H} + \frac{9+6q}{n_H} = \frac{1}{\lfloor(k-1)\rho/3\rfloor-1} + \frac{12+6q}{n_H} \\
&= \frac{1}{\lfloor(k-1)\rho/3\rfloor-1} + \frac{12+6\log(n_H)}{n_H}
\end{aligned}$$

This implies the conditions related to σ_Δ and t_Δ are satisfied. From Lemma 1, it follows that with probability $1-2\delta$ GOPA is (ϵ,δ)-differentially private, or in short GOPA is $(\epsilon, 3\delta)$-differentially private. □

A.4 Matching the Utility of the Centralized Gaussian Mechanism

From the above theorems, we can now obtain a simple corollary which precisely quantifies the amount of independent and pairwise noise needed to achieve a desired privacy guarantee depending on the topology.

Proof of Corollary 1. In the centralized (trusted curator) setting, the standard centralized Gaussian mechanism [33] (Theorem A.1 therein) states that in order for the noisy average $(\frac{1}{n}\sum_{u\in U}X_u)+\eta$ to be (ε',δ')-DP for some $\varepsilon',\delta' \in (0,1)$, the variance of η needs to be:

$$\sigma_{gm}^2 = \frac{c^2}{(\varepsilon' n)^2}. \tag{31}$$

where $c^2 > 2\log(1.25/\delta')$.

Based on this, we let the independent noise η_u added by each user in GOPA to have variance

$$\sigma_\eta^2 = \frac{n^2}{n_H}\sigma_{gm}^2 = \frac{c^2}{(\varepsilon')^2 n_H}, \tag{32}$$

which, for the approximate average $\hat{X}^{avg}$, gives a total variance of:

$$Var\Big(\frac{1}{n_H}\sum_{u\in U^H}\eta_u\Big) = \frac{1}{n_H^2}n_H\sigma_\eta^2 = \frac{c^2}{(\varepsilon' n_H)^2}. \tag{33}$$

We can see that when $n_H = n$ (no malicious user), Equation (33) exactly corresponds to the variance required by the centralized Gaussian mechanism in Equation (31), hence GOPA will achieve the same utility. When there are malicious users, each honest user needs to add a factor n/n_H more noise to compensate for the fact that malicious users can subtract their own inputs and independent noise terms from $\hat{X}^{avg}$. This is consistent with previous work on distributed noise generation under malicious parties [64].

Now, given some $\kappa > 0$, let $\sigma_\Delta^2 = \kappa\sigma_\eta^2$ if G is the complete graph, $\sigma_\Delta^2 = \frac{1}{\lfloor(k-1)\rho/3\rfloor-1} + (12 + 6\log(n_H))/n_H$ for the random k-out graph, and $\sigma_\Delta^2 = \kappa n_H^2\sigma_\eta^2/3$ for an arbitrary connected G^H. In all cases, the value of θ in Theorems 1, 2 and 3 after plugging σ_Δ^2 gives

$$\theta = \frac{\varepsilon^2}{c^2} + \frac{\varepsilon^2}{\kappa c^2} = \frac{(\kappa+1)\varepsilon^2}{\kappa c^2}.$$

$n = 100$	$\rho = 1$	$k = 3$	$\sigma_\Delta = 60.8$
		$k = 5$	$\sigma_\Delta = 41.3$
	$\rho = 0.5$	$k = 20$	$\sigma_\Delta = 26.8$
		$k = 30$	$\sigma_\Delta = 17.2$
$n = 1000$	$\rho = 1$	$k = 5$	$\sigma_\Delta = 63.4$
		$k = 10$	$\sigma_\Delta = 41.1$
	$\rho = 0.5$	$k = 20$	$\sigma_\Delta = 45.4$
		$k = 30$	$\sigma_\Delta = 27.3$
$n = 10000$	$\rho = 1$	$k = 10$	$\sigma_\Delta = 54.6$
		$k = 20$	$\sigma_\Delta = 34.7$
	$\rho = 0.5$	$k = 20$	$\sigma_\Delta = 55.5$
		$k = 40$	$\sigma_\Delta = 28.4$

Table 2: Examples of admissible values for k and σ_Δ, obtained by numerical simulation, to ensure (ϵ, δ)-DP with trusted curator utility for $\varepsilon = 0.1$, $\delta' = 1/n_H^2$, $\delta = 10\delta'$.

By setting $\varepsilon = \varepsilon'$, Equation (3) of Theorem 1 implies:

$$\varepsilon \geq \frac{(\kappa+1)\varepsilon^2}{2\kappa c^2} + \sqrt{\frac{(\kappa+1)}{\kappa}}\frac{\varepsilon}{c}$$

For $d^2 = \frac{\kappa}{\kappa+1}c^2$ we can rewrite the above as

$$\varepsilon \geq \frac{\varepsilon^2}{2d^2} + \frac{\varepsilon}{d}.$$

Since $\epsilon \leq 1$, this is satisfied if $d - \frac{\varepsilon}{2d} \geq 1$ and in turn when $d \geq 3/2$, or equivalently when $c \geq \frac{3}{2}\sqrt{\frac{\kappa+1}{\kappa}}$. Now analyzing the inequality in Equation (4) we have:

$$\begin{aligned}
\Big(\varepsilon - \frac{(k+1)\varepsilon^2}{2\kappa c^2}\Big)^2 &\geq 2\log(2/\delta\sqrt{2\pi})\Big(\frac{\varepsilon^2}{c^2} + \frac{\varepsilon^2}{\kappa c^2}\Big) \\
\varepsilon^2 + \frac{(\kappa+1)^2\varepsilon^4}{4\kappa^2 c^4} - \frac{(\kappa+1)\varepsilon^3}{\kappa c^2} &\geq 2\log(2/\delta\sqrt{2\pi})\Big(\frac{(\kappa+1)\varepsilon^2}{\kappa c^2}\Big) \\
\frac{1}{2}\Big(\frac{\kappa c^2}{\kappa+1} + \frac{(\kappa+1)\varepsilon^2}{4\kappa c^2} - \varepsilon\Big) &\geq \log(2/\delta\sqrt{2\pi}).
\end{aligned}$$

Again denoting $d^2 = \frac{\kappa}{\kappa+1}c^2$ we can rewrite the above as

$$\frac{1}{2}\Big(d^2 + \frac{\varepsilon^2}{4d^2} - \varepsilon\Big) \geq \log(2/\delta\sqrt{2\pi}).$$

For $d \geq 3/2$ and $\varepsilon \leq 1$, the derivative of $d^2 + \frac{\varepsilon^2}{4d^2} - \varepsilon$ is positive, so $d^2 + \frac{\varepsilon^2}{4d^2} - \varepsilon > d^2 - 8/9$. Thus, we only require $d^2 \geq 2\log(1.25/\delta)$. Therefore Equation (4) is satisfied when:

$$\frac{\kappa}{\kappa+1}\log(1.25/\delta') \geq \log(1.25/\delta),$$

which is equivalent to

$$\delta \geq 1.25\Big(\frac{\delta'}{1.25}\Big)^{\frac{\kappa}{\kappa+1}}.$$

The constant 3.75 instead of 1.25 for the random k-out graph case is because Theorem 3 guarantees $(\epsilon, 3\delta)$-DP instead of (ϵ, δ) in Theorems 1 and 2. □

A.5 Smaller k and σ_Δ^2 via Numerical Simulation

For random k-out graphs, the conditions on k and σ_Δ^2 given by Theorem 3 are quite conservative. While we are confident that they can be refined by resorting to tighter approximations in our analysis in Section A.3, an alternative option to find smaller, yet admissible values for k and σ_Δ^2 is to resort to numerical simulation.

Given the number of users n, the proportion of honest nodes ρ and a value for k, we implemented a program that generates a random k-out graph, checks if the subgraph G^H of honest users is connected, and if so finds a suitable spanning tree for G^H and computes the corresponding value for $t_\Delta^\top t_\Delta$ needed by our differential privacy analysis (see for instance Appendix A.2). From this, we can in turn deduce a sufficient value for σ_Δ^2 using Corollary 1.

Table 2 gives examples of values obtained by simulations for various values of n, ρ and several choices for k. In each case, the reported σ_Δ corresponds to the *worst-case* value required across 10^5 random runs, and the chosen value of k was large enough for G^H to be connected in *all* runs. This was the case even for slightly smaller values of k. Therefore, the values reported in Table 2 can be considered safe to use in practice.

Appendix B Cryptographic Framework

In this appendix, we provide an extended overview of the cryptographic notions involved in Section 5, and give more details about our verification protocol. First, we give deeper intuitions on commitments and ZKPs in Appendices B.1 and B.2. Next, using these building blocks, we present ZKPs on the statistical distributions of uniform and Gaussian random variables in Appendix B.3. Finally, in Appendix B.4, we show how these tools can be used to obtain the verification protocol of GOPA. For simplicity, we sometimes abstract away some of the technical cryptographic details, but refer to the corresponding bibliography for a more advanced treatment.

Public bulletin board. We implement the publication of commitments and proofs using a public bulletin board so that any party can verify the validity of the protocol, avoiding the need for a trusted verification entity. Users sign their messages so they cannot deny them. More general purpose distributed ledger technology such as in Bitcoin transactions [60] could be used here, but we aim at an application-specific, light-weight and hence more scalable solution.

B.1 Commitment Schemes

We start by defining formally a commitment and its properties. For clarity, we will use bold variables to denote commitments.

Definition 2 (Commitment Scheme)**.** *A commitment scheme consists of a pair of (computationally efficient) algorithms $(Setup, Com)$. The setup algorithm $Setup$ is executed once, with randomness t as input, and outputs a tuple $\Theta \leftarrow Setup(t)$, which is called the set of parameters of the scheme. The algorithm Com with parameters Θ, denoted Com_Θ, is a function $Com_\Theta : \mathcal{M}_\Theta \times \mathcal{R}_\Theta \to \mathcal{C}_\Theta$, where $\mathcal{M}_\Theta$ is called the message space, $\mathcal{R}_\Theta$ the randomness space, and $\mathcal{C}_\Theta$ the commitment space. For a message $m \in \mathcal{M}_\Theta$, the algorithm draws $r \in \mathcal{R}_\Theta$ uniformly at random and computes commitment $\mathbf{c} \leftarrow Com_\Theta(m, r)$.*

The security of a commitment scheme typically depends on t not being biased, in particular, it must be hard to guess non-trivial information about t.

We now define some key properties of commitments.

Property 1 (Hiding Property)**.** *A commitment scheme is* hiding *if, for all secrets $x \in \mathcal{M}_\Theta$ and given that r is chosen uniformly at random from $\mathcal{R}_\Theta$, the commitment $\mathbf{c_x} = Com_\Theta(x, r)$ does not reveal any information about x.*

Property 2 (Binding Property)**.** *A commitment is* binding *if there exists no computationally efficient algorithm $\mathcal{A}$ that can find $x_1, x_2 \in \mathcal{M}_\Theta$, $r_1, r_2 \in \mathcal{R}_\Theta$ such that $x_1 \neq x_2$ and $Com_\Theta(x_1, r_1) = Com_\Theta(x_2, r_2)$.*

Property 3 (Homomorphic Property)**.** *A* homomorphic *commitment scheme is a commitment scheme such that $\mathcal{M}_\Theta$, $\mathcal{R}_\Theta$ and $\mathcal{C}_\Theta$ are abelian groups, and for all $x_1, x_2 \in \mathcal{M}_\Theta$, $r_1, r_2 \in \mathcal{R}_\Theta$ we have*

$$Com_\Theta(x_1, r_1) + Com_\Theta(x_2, r_2) = Com_\Theta(x_1 + x_2, r_1 + r_2).$$

Please note that the three occurrences of the '$+$' sign in the above definition are operations in three difference spaces, and hence may have different definitions and do not necessarily correspond to normal addition of numbers.

Pedersen commitments. For our protocol we use the Pedersen commitment scheme, described in [62]. For a given cyclic group $\mathbb{G}$ of large prime order q, $Setup$ is a function that draws uniformly

and independently at random from $\mathbb{G}$ a pair of elements g and h which form the set of parameters $\Theta = (g, h)$ of the scheme. Additionally, $\mathcal{M}_\Theta = \mathcal{R}_\Theta = \mathbb{Z}_q$, and $\mathcal{C}_\Theta = \mathbb{G}$. We will refer to g and h as *bases*. The commitment function Com_Θ is defined as

$$\begin{aligned} Com_\Theta &: \mathbb{Z}_q \times \mathbb{Z}_q \to \mathbb{G} \\ Com_\Theta(x, r) &= g^x \cdot h^r, \end{aligned} \tag{34}$$

where $(\cdot)$ is the modular product of group $\mathbb{G}$, x is the secret and r is the randomness. Pedersen commitments are homomorphic (in particular, $Com_\Theta(x + y, r + s) = Com_\Theta(x, r) \cdot Com_\Theta(y, s)$), unconditionally hiding, and computationally binding under the Discrete Logarithm Assumption, which we informally describe below.

Assumption 1 (Discrete Logarithm Assumption)**.** *Let $\mathbb{G}$ be a cyclic multiplicative group of large prime order q, and g and h two elements chosen independently and uniformly at random from $\mathbb{G}$. Then, there exists no probabilistic polynomial time algorithm $\mathcal{A}$ that takes as input the tuple $(\mathbb{G}, q, g, h)$ and outputs a value b such that $P(g^b = h)$ is significant on the bit-size of q.*

The DLA is a standard assumption widely used in production environments. A more detailed description can be found in Chapter 7 of [54].

The Pedersen commitment scheme can be efficiently implemented with elliptic curves, as described and benchmarked in [39]. To generate a common Pedersen scheme in our adversary model, the generation of the unbiased random input t for $Setup$ can be done as described in Section 4.4 of [18]. We provide a concrete method for this in Appendix C.

B.2 Zero Knowledge Proofs

On top of Pedersen commitments, we use a family of techniques called Zero Knowledge Proofs (ZKPs), first proposed by [43]. In these proofs, a party P called the Prover, convinces another party V, the Verifier, about a statement. For our scope and informally speaking, ZKPs[3]

- allow every P to successfully prove true a statement (completeness),
- allow every V to discover with arbitrarily large probability any attempt to prove a false statement (soundness),
- guarantee that by performing the proof, no information about the knowledge of P other than the proven statement is revealed (zero knowledge).

Importantly, the zero knowledge property of our proofs does not rely on any computational hardness assumption.

Honest-verifier interactive ZKPs. In the following sections, we describe how our ZKPs can be be performed as an interactive protocol between a Prover P and a honest Verifier V whose only task in the conversation is to generate unbiased random numbers, that we will call *challenges*, and which P must not be able to predict. In these proofs, the zero knowledge property holds only if V is honest and does not try to generate biased challenges in order to guess private values of P. Such model does not apply to our malicious adversary model. However, this can be easily overcome in most cases with a generic transformation of the proofs that makes them suitable in adversarial settings, called the Fiat-Shamir heuristic, which we describe at the end of Appendix B.3. Additionally, for a particular case of a proof in the beginning of Appendix B.3 which, as we will see, cannot be made non-interactive, we provide an efficient interactive method to generate unbiased challenges as an honest verifier would do in Appendix C.

B.2.1 Zero Knowledge Proofs: Building Blocks

We now briefly review a number of ZKP building blocks present in literature and necessary for our verification protocol. We illustrate the basic principles for a selection of relevant ZKPs, describing their protocol and complexity, and refer to the literature for others.

[3] Strictly speaking, the proofs we will use are called *arguments*, as the soundness property relies on the computational boundedness of the Prover P through the DLA described above, but as for general reference to the family of techniques we use the term *proofs*.

We base our proofs in a Pedersen commitment with parameters (g, h) chosen from $\mathbb{G}$ as described above. For some building blocks, we use multiple pairs of bases as parameters, where bases belonging to the same pair Θ are always assumed to be chosen independently at random from $\mathbb{G}$. We denote by $r \leftarrow_R S$ the operation of generating a random number r uniformly distributed over the set S, and by $a =_? b$ the action of verifying the equality of a and b. Between elements of $\mathbb{G}$, products and exponentiations are implicitly modulo q.

When discussing complexity, the dominating computations are always discrete exponentiations in $\mathbb{G}$. We will refer to them just as exponentiations. For a ZKP, we call the *size of a proof* the sum of all the values (numbers, commitments, challenges, etc.) exchanged between P and V.

Proof of equality. This first proof allows a prover P to prove that he committed to the same private value x in two different commitments $\mathbf{c_x} = g_1^x \cdot h_1^r$ and $\mathbf{c'_x} = g_2^x \cdot h_2^{r'}$. The pairs of bases (g_1, h_1) and (g_2, h_2) might be different. This proof is a standard generalization of the one described in [24] for single bases.

The protocol goes as follows:

1. P generates $a, b, c \leftarrow_R \mathbb{Z}_q$, computes $\mathbf{c_{ab}} \leftarrow_R g_1^a h_1^b$ and $\mathbf{c_{ac}} \leftarrow g_2^a h_2^c$ and sends $(\mathbf{c_{ab}}, \mathbf{c_{ac}})$ to V.
2. V draws a challenge $t \leftarrow_R \mathbb{Z}_q$ and sends it to P.
3. P computes $d \leftarrow a + xt \pmod q$, $e \leftarrow b + rt \pmod q$ and $f \leftarrow c + r't \pmod q$ and sends (d, e, f) to V.

At any time, to verify the proof $(\mathbf{c_{ab}}, \mathbf{c_{ac}}, t, d, e, f)$, V checks that

$$g_1^d h_1^e =_? \mathbf{c_{ab}}(\mathbf{c_x})^t \quad \text{and} \quad g_2^d h_2^f =_? \mathbf{c_{ac}}(\mathbf{c'_x})^t.$$

If the equality holds, P proves the statement. We denote this proof as

$$\mathbf{ZKPEq}(\mathbf{c_x}, \mathbf{c'_x})\left\{(x, r, r') : \mathbf{c_x} = g_1^x h_1^r \wedge \mathbf{c'_x} = g_2^x h_2^{r'}\right\},$$

where we use the convention that values outside the curly braces are public, while values private to P and the proven statement are inside. The proof requires the computation of 4 exponentiations by P , 6 exponentiations by V, and its size is the sum of the sizes of the 6 values exchanged in the conversation between P and V.

Proof of linear relation. This is a key proof that will be used several times in our verification procedure. Let $\bar{x} = (x_i)_{i=1}^k$ for some $k > 0$ be a vector of private values, $\bar{\mathbf{c}} = (\mathbf{c_i})_{i=1}^k$ a public vector of commitments such that $\mathbf{c_i}$ is a commitment of x_i, $\bar{a} = (a_i)_{i=1}^k$ a public vector of coefficients, and b a public scalar. The goal of P is to prove the equality $\langle \bar{x}, \bar{a} \rangle = b \pmod q$, where $\langle \cdot, \cdot \rangle$ is the inner product. The proof is simple: first, note that, by the homomorphic property, $\mathbf{c_L} = \prod_{i=1}^k (\mathbf{c_i})^{(a_i)}$ is a valid commitment of $\langle \bar{x}, \bar{a} \rangle$. Then, P and V just perform $\mathbf{ZKPEq}(\mathbf{c_L}, \mathbf{c_b})$ for $\mathbf{c_b} = g^b$ (the base h for $\mathbf{c_b}$ is not necessary as we use randomness equal to 0). We denote the proof as

$$\mathbf{ZKPLinear}(\bar{\mathbf{c}}, \bar{a}, b)\left\{\left(\bar{x}, \bar{r} = (r_i)_{i=1}^k\right) : \bigwedge_{i=1}^k \mathbf{c_i} = g^x h^{r_i} \wedge \langle \bar{a}, \bar{x} \rangle = b \pmod q\right\}.$$

To perform the proof, the amount of computations done by P is dominated by $3k + 1$ exponentiations to compute the vector $\bar{\mathbf{c}}$ and the other commitments $\mathbf{c_L}$ and $\mathbf{c_b}$. Both P and V perform one execution of **ZKPEq**. V also has to compute $\mathbf{c_L}$ and $\mathbf{c_b}$ by performing $k+1$ exponentiations. The computational complexity is then $O(k)$ exponentiations for both parties. The proof size and therefore the load of the network is also $O(k)$ values, which for P, can be split into several messages if components of $\bar{\mathbf{c}}$ are committed in different moments. Note that, with this proof, users can already prove the correctness of Properties (5) and (6) for the verification of GOPA.

Proof that a secret lies in a given range (*range proof*). Here, the statement to prove is that, for a positive integer $M < q - 1$ and a commitment $\mathbf{c_x}$, P knows the secret x inside $\mathbf{c_x}$, and that it lies in the range $[0, M]$. Among the various constructions that exist for range proofs, we use the result by [18] for intervals of the form $[0, 2^l - 1]$. Then, our proof can be performed with 2 of these proofs,

where P proves that $x \in [0, 2^l - 1]$ and that $M - x \in [0, 2^l - 1]$ (this involves another commitment for $M - x$ and a linear proof of this relation) for $l = \lfloor \log_2 M \rfloor + 1$. We denote the proof by

$$\mathbf{ZKPRange}(\mathbf{c_x}, [0, M])\big\{(x, r) : \mathbf{c_x} = g^x h^r \wedge x \in [0, M]\big\}.$$

It has a computational cost of $O(\log_2 M)$ exponentiations for both P and V and a proof size of $O(\log_2 \log_2 M)$. With this proof, the verification of Property (8) can be done.

The verification of Property (7) is more elaborate as it involves proving that a value has been drawn from a prescribed statistical distribution. We describe this proof in Appendix B.3, but first introduce additional building blocks that will be needed for this proof.

Proof of product. In this proof, P wants to prove that for three secrets a, b, d committed in $\mathbf{c_a} = g^a h^{r_1}$, $\mathbf{c_b} = g^b h^{r_2}$ and $\mathbf{c_d} = g^d h^{r_3}$, it holds that $ab = d \pmod q$. This proof can be done in a straightforward way based on **ZKPEq**. We use the fact that, by properties of cyclic groups, $\mathbf{c_a}$ is not only a commitment but also a base of $\mathbb{G}$, and that $\mathbf{c_d} = (\mathbf{c_a})^b h^{(r_2 - b r_a)}$, which is a valid commitment for b using bases $(\mathbf{c_a}, h)$. Then we use $\mathbf{ZKPEq}(\mathbf{c_b}, \mathbf{c_d})$ with the pairs of bases (g, h) and $(\mathbf{c_a}, h)$. If P does not know the commitment of a product of ab hidden in $\mathbf{c_d}$ it would not pass the proof. We denote this proof as

$$\begin{aligned}\mathbf{ZKPProd}(\mathbf{c_a}, \mathbf{c_b}, \mathbf{c_d})\big\{(a, r_1, b, r_2, d, r_3) : \mathbf{c_a} &= g^a h^{r_1} \wedge \mathbf{c_b} = g^b h^{r_2} \\ &\wedge \mathbf{c_d} = g^d h^{r_3} \wedge ab = d \pmod q\big\}.\end{aligned}$$

The proof, first proposed in [40], has the same complexity as **ZKPEq** plus the computation of an extra commitment (2 exponentiations).

Proof of OR-composition. This proof provides the possibility of constructing a ZKP on a statement that is the disjunction of statements of other (already constructed) ZKPs. Concretely, given the proofs $\mathbf{ZKP_1}\{(\bar{v}) : S_1(\bar{v})\}$, and $\mathbf{ZKP_2}\{(\bar{v}) : S_2(\bar{v})\}$ where $\bar{v}$ is a vector of private values and S_1 and S_2 are the verifiable statements, one can construct the proof

$$\mathbf{ZKPOr}\big\{(\bar{v}) : S_1(\bar{v}) \vee S_2(\bar{v})\big\}.$$

The proof is described in [27] and [63]. The complexity of **ZKPOr** is the sum of the complexities of $\mathbf{ZKP_1}$ and $\mathbf{ZKP_2}$. Not exclusively but often, we will use the construction to prove that a secret b is a bit, i.e. that $b \in \{0, 1\}$. The instantiation is described in [57](pages 66-67 therein) and requires $O(1)$ computations for P and V and as well as $O(1)$ proof size complexity.

Proof of modular sum. Here, P wants to prove that, for a public modulus $M < q/2$, a public value $t \in [0, M - 1]$ and two secrets $x, z \in [0, M - 1]$, the relation $x = z + t \mod M$ is satisfied. Let $\mathbf{c_x}$ and $\mathbf{c_z}$ be the commitments of x and z respectively. For the proof, we perform $\mathbf{ZKPRange}(\mathbf{c_x}, [0, M - 1])$, $\mathbf{ZKPRange}(\mathbf{c_z}, [0, M - 1])$, and we use an auxiliary commitment of a bit b which we prove to be in $\{0, 1\}$. Then, we prove the arithmetic relation $x = z + t - bM \pmod q$ with **ZKPLinear**. Because of their range, we have that $z + t < q$, hence the modulus q does not interfere and the linear equality holds if and only if the modular sum is satisfied. We denote this proof as

$$\begin{aligned}\mathbf{ZKPMod}(\mathbf{c_x}, \mathbf{c_z}, M, t)\big\{(x, r_1, z, r_2) : \mathbf{c_x} &= g^x h^{r_1} \wedge x \in [0, M - 1] \\ &\wedge \mathbf{c_z} = g^z h^{r_2} \wedge z \in [0, M - 1] \\ &\wedge x = z + t \mod M\big\}.\end{aligned}$$

This proof is inspired by the more general proof of modular sum proposed in [20]. Its complexity is given by the composition of the proofs mentioned above, where the dominant term is in the range proofs, and amounts to $O(\log_2 M)$ exponentiations in the computation by both P and V, and $O(\log_2 \log_2 M)$ in the published proof size.

Non-interactive ZKPs. There is a standard method to transform honest verifier interactive ZKPs into a non-interactive protocol through the widely used Fiat-Shamir heuristic [37]. Roughly speaking, the method replaces the interactive role of a verifier V by a cryptographic hash function (in theory,

a random oracle) which generates our publicly verifiable challenges. Using this heuristic, P can compute the proof by itself without compromising the soundness property, and publish it in the bulletin board. The complexity of generating and verifying the proof is not changed by the transformation, and the proof is established by the fictional interactive conversation between P and V. Once P has published the proof in the bulletin board, any party can check its correctness.

B.3 Zero Knowledge Proofs for Uniform and Gaussian Random Variables

Given the building blocks in Appendix B.2, we can construct a proof that a random variable follows a uniform or a Gaussian distribution.

Proof of uniform distribution. We prove that a secret random variable y is uniformly distributed over the interval $[0, M-1]$ for $M < q/2$. As we are not proving any statement for a particular fixed value but of a distribution, it is intuitive that we sample the value while the proof is generated. Essentially, we use $\textbf{ZKPMod}(\mathbf{c_y}, \mathbf{c_z}, M, t)$, and provide a protocol such that t guarantees the distribution is not biased and z guarantees the secrecy of y. By the properties of modular sum, we can ensure that y is uniformly distributed. The secure proof goes as follows:

1. P draws $z \leftarrow_R [0, M-1]$, computes $\mathbf{c_z} = g^z h^r$ for some $r \leftarrow_R \mathbb{Z}_q$ and sends $\mathbf{c_z}$ to V.
2. V draws $t \leftarrow_R [0, M-1]$ (as described in Appendix C) and sends it to P.
3. P computes $y \leftarrow z + t \mod M$ and $\mathbf{c_y} \leftarrow g^y h^{r'}$ for some $r' \leftarrow_R \mathbb{Z}_q$, and sends $\mathbf{c_y}$ to V.

After this, P and V perform **ZKPMod** as mentioned above. We denote this proof as

$$\textbf{ZKPUniform}(\mathbf{c_y}, [0, M-1])\Big\{(y, r') : \mathbf{c_y} = g^y h^{r'} \wedge y \sim \mathcal{U}([0, M-1])\Big\}.$$

The complexity of **ZKPUniform** remains of the same order as **ZKPMod**, i.e. $O(\log_2 M)$ computation and $O(\log_2 \log_2 M)$ in the published proof size.

Note that this proof cannot be made non-interactive, as a malicious user can sample locally many random values and pick the one he likes, biasing the distribution. We need to make sure all users generate uniformly distributed numbers in the first attempt and do not have possibility to pick and reject numbers as they see fit. We provide in Appendix C a decentralized method to generate unbiased random challenges, which makes the proving user to "commit" to every generation of a random number. This approach only requires $O(1)$ additional computations and communication cost per user, so the complexity of **ZKPUniform** remains of the same order.

Proof of Gaussian distribution. In our algorithm, every user u needs to generate a Gaussian distributed number η_u, which he does not publish, but for which we need to verify that it is generated correctly, as otherwise a malicious user could bias the result of the algorithm.

Proving the generation of a Gaussian distributed random number is more involved. We will start from an integer y', which is only known to a party P, for which P has published a commitment $Com_\Theta(y')$ and for which the other agents know it has been generated uniformly randomly from an interval $[0, M-1]$ (for large M), as can be done with **ZKPUniform**. Then, P will compute x' such that $((2y'+1)/M) - 1 = \text{erf}(x'/\sqrt{2})$. We know that x' is normally distributed. The main task is then to provide a ZKP that $y = \text{erf}(x)$ for $y = ((2y'+1)/M) - 1$ and $x = x'/\sqrt{2}$.

The error function relates its input and output in a way that cannot be expressed with additive, multiplicative or exponential equations. We therefore approximate erf using a converging series. In particular, we will rely on the series

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \sum_{l=0}^{\infty} \frac{(-1)^l x^{2l+1}}{l!(2l+1)}. \tag{35}$$

As argued in [26], this series has two major advantages. First, it only involves additions and multiplications, while other known series converging to $\text{erf}(x)$ often include factors such as $\exp(-x^2/2)$ which would require additional evaluations and proofs. Second, it is an alternating series, which means we can determine more easily in advance how many terms we need to evaluate to achieve a given precision.

Nevertheless, Equation (35) converges slowly for large x. It is more efficient to prove either that

$$y = \text{erf}(x) \quad \text{or} \quad 1 - y = \text{erfc}(x),$$

as for $\text{erfc}(x) = 1 - \text{erf}(x)$ there exist good approximations requiring only a few terms for large x. An example is the asymptotic expansion

$$\text{erfc}(x) = \frac{e^{-x^2}}{x\sqrt{\pi}} S_{\text{erfc}}(x) + R_L(x), \tag{36}$$

where

$$S_{\text{erfc}}(x) = \sum_{l=0}^{L-1} \frac{(-1)^l (2l-1)!!}{(2x^2)^l} \tag{37}$$

with $l!! = 1$ for $l < 1$ and $(2l-1)!! = \prod_{i=1}^{l}(2i-1)$. This series diverges, but if x is sufficiently large then the remainder

$$R_L(x) \leq \frac{2}{\sqrt{\pi}} \frac{(2L-1)!!}{(2x^2)^L} \tag{38}$$

after the first L terms is sufficiently small to be neglected. So the prover could prove either part of the disjunction with **ZKPOr** depending on whether the erf or erfc approximations achieve sufficient precision. In particular, for a fixed x, Equation (38) is minimal with $L \approx x^2/2$, so given x we can achieve an error of

$$\begin{aligned}
E_{\text{erfc}}(x) &\approx \frac{2}{\sqrt{\pi}} \frac{(2\lfloor x^2/2 \rfloor - 1)!!}{(2x^2)^{\lfloor x^2/2 \rfloor}} \\
&= \frac{2}{\sqrt{\pi}} \frac{(2\lfloor x^2/2 \rfloor)!}{\lfloor x^2/2 \rfloor! 2^{\lfloor x^2/2 \rfloor} (2x^2)^{\lfloor x^2/2 \rfloor}} \\
&\approx \frac{2}{\sqrt{\pi}} \frac{(2\lfloor x^2/2 \rfloor/e)^{2\lfloor x^2/2 \rfloor} (2\lfloor x^2/2 \rfloor)^{1/2}}{(\lfloor x^2/2 \rfloor/e)^{\lfloor x^2/2 \rfloor} (\lfloor x^2/2 \rfloor)^{1/2} 2^{\lfloor x^2/2 \rfloor} (2x^2)^{\lfloor x^2/2 \rfloor}} \\
&= \frac{2}{\sqrt{\pi}} \frac{(\lfloor x^2/2 \rfloor)^{\lfloor x^2/2 \rfloor} 2^{1/2}}{e^{\lfloor x^2/2 \rfloor} (x^2)^{\lfloor x^2/2 \rfloor}} \leq \frac{\sqrt{8}}{\sqrt{\pi}(2e)^{\lfloor x^2/2 \rfloor}}.
\end{aligned}$$

Approximating $\text{erfc}(x)$ involves approximating $\exp(-x^2)$. The common series $\exp(z) = \sum_{i=0}^{\infty} z^i/i!$ is known to converge quickly. Even if its terms first go up until $z < i$, for larger z where this could slow down convergence one can simply divide z be a constant a (maybe conveniently a power of 2), approximate $\exp(z/a)$ and then compute $(\exp(z/a))^a$, which can be done efficiently. We omit the details here and focus in the following on the more difficult erf function.

In absolute value, terms of the erf approximation never get larger than

$$\begin{aligned}
\left| \frac{(-1)^l x^{2l+1}}{l!(2l+1)} \right| &\leq \frac{x^{2\lfloor x^2 \rfloor + 1}}{\lfloor x^2 \rfloor! (2x^2 - 1)} \\
&\approx \frac{x^{2x^2+1}}{\sqrt{2\pi x^2}(x^2/e)^{x^2}(2x^2-1)} \\
&= \frac{x}{\sqrt{2\pi} x e^{-x^2}(2x^2-1)} = \frac{e^{x^2}}{\sqrt{2\pi}(2x^2-1)} \\
&< \frac{e^{x^2}}{\sqrt{8\pi x^2}}
\end{aligned} \tag{39}$$

Evaluating a term of one of the series (Equation (35) or Equation (37)) is possible by addition, multiplication and a division. We have described above building blocks for providing ZKPs of additions (**ZKPLinear**) and multiplications (**ZKPProd**). For the multiplication, please note that one can compute x^l by $x^l = x^{l-1}x$, reusing the result of earlier multiplications. For the division, proving that $a/b = c$, where a and c are private, is possible by proving that $-|b|/2 \leq a - bc \leq |b|/2$, which involves a range proof. The several range proofs for the several terms can be combined as indicated in the description of **ZKPRange**.

In the following, we will explicitly consider fixed precision representations. The implied rounding does not cause major problems for several reasons. First, the Gaussian distribution is symmetric, and hence the probability of rounding up and rounding down is exactly the same, making the rounding error a zero-mean random variable. Second, discrete approximations of the Gaussian mechanism such as binomial mechanisms have been studied and found to give similar guarantees as the Gaussian mechanism [2]. Third, we can require the cumulated rounding error to be an order of magnitude smaller than the standard deviation of the noise we are generating, so that any deviation due to rounding has negligible impact.

We will use a fixed precision for all numbers (except for small integer constants), and we will represent numbers as multiples of ψ. We assume $1/\psi M$ to be an integer, so as the variable x is a multiple of $1/M$, we can represent x and x^2 without rounding. Let

$$t_i = \frac{x^{2i+1}}{i!}.$$

We can compute recursively

$$t_{i+1} = t_i \frac{x^2}{i+1},$$

where $t_0 = x$. The multiplication with x^2 and the division by i increase the relative error by less than ψ. If we evaluate the first L terms of the series, the relative error due to rounding is thus at most $L\psi$. Combining this with Equation (39) implies that the maximal absolute error of a term is $L\psi e^{x^2}/\sqrt{8\pi}x^2$, and the total error after summing these terms is

$$E_{\text{erf}}^{round} = \frac{L^2 \psi e^{x^2}}{\sqrt{8\pi}x^2}. \tag{40}$$

Suppose we want to achieve an approximation where the error on y as a function of x is at most B. It is natural to set $M \approx 1/B$. We require that the rounding error is at most $B/2$ and the approximation error is at most $B/2$. Then, if

$$\frac{B}{2} \geq \frac{\sqrt{8}}{\sqrt{\pi}(2e)^{\lfloor x^2/2 \rfloor}} \geq E_{\text{erfc}}(x),$$

the prover will use the erfc series, else the erf series. As the erf series more strongly constrains our design choices, we will focus here on the erf case (the erfc case can be analyzed using the same principles). We hence assume that

$$\frac{\sqrt{8}}{\sqrt{\pi}(2e)^{\lfloor x^2/2 \rfloor}} \geq \frac{B}{2},$$

which implies

$$\frac{\sqrt{32}}{B\sqrt{\pi}} \geq (2e)^{\lfloor x^2/2 \rfloor},$$

from which

$$\frac{1}{2}\log\left(\frac{32}{\pi}\right) - \log(B) \geq \left\lfloor \frac{x^2}{2} \right\rfloor \log(2e).$$

This implies

$$\frac{1.161 - \log(B)}{1.693} \geq \frac{x^2}{2} - 1$$

and

$$\sqrt{\frac{2.854 - \log(B)}{0.846}} \geq x. \tag{41}$$

The series in Equation (35) is an alternating series too, so to reach an error smaller than $B/2$, it is sufficient to truncate the series when terms get smaller than $B/2$ in absolute value. In particular, we need L terms with

$$\frac{2x^{2L+1}}{\sqrt{\pi}L!(2L+1)} \leq \frac{B}{2}.$$

Using Stirling's approximation, this means

$$\frac{2x^{2L+1}}{\sqrt{\pi}\sqrt{2\pi L}(L/e)^L(2L+1)} \leq \frac{B}{2}.$$

Taking logarithms, we get:

$$\log(2) + (2L+1)\log(x) - \log(\pi\sqrt{2}) - (L+1/2)\log(l) - L + \log(2L+1) \leq \log(B) - \log(2).$$

For $L \geq 1$, it holds that $l \geq 2\log(2) + \log(2L+1) - \log(\pi\sqrt{2})$, so the above inequality is satisfied if

$$(2L+1)\log(x) - (L+1/2)\log(L) \leq \log(B),$$

which is equivalent to

$$(2L+1)\log(x/\sqrt{L}) \leq \log(B),$$

or, written differently:

$$(2L+1)\log\left(1 - \frac{\sqrt{L}-x}{\sqrt{L}}\right) \leq \log(B).$$

As $\log(1+\alpha) \leq \alpha$, the above inequality is satisfied if

$$-(2L+1)\frac{\sqrt{L}-x}{\sqrt{L}} \leq \log(B).$$

Further approximating, this is satisfied if

$$2L\frac{\sqrt{L}-x}{\sqrt{L}} + \log(B) \geq 0,$$

or equivalently

$$2L - 2\sqrt{L}x + \log(B) \geq 0.$$

It follows that we need

$$\sqrt{L} \geq \frac{2x + \sqrt{4x^2 - 8\log(B)}}{4} = \frac{x + \sqrt{x^2 - 2\log(B)}}{2}.$$

Substituting the worst case value of x from Equation (41), we get

$$L \geq \frac{1}{4}\left(\sqrt{\frac{2.854 - \log(B)}{0.846}} + \sqrt{\frac{2.854 - 2.692\log(B)}{0.846}}\right)^2. \tag{42}$$

From Equation (40), we can now also determine what precision is needed. In particular:

$$E_{\text{erf}}^{round} = \frac{L^2\psi e^{x^2}}{\sqrt{8\pi}x^2} \leq \frac{B}{2}$$

is satisfied if

$$\psi \leq \frac{\sqrt{2\pi}Bx^2}{L^2e^{x^2}} = O\left(\frac{B^2}{\log(B)}\right).$$

Typically, one would like the total error (due to approximation and rounding) to be negligible with respect to the standard deviation σ_η, so one could choose $B = \sigma_\eta/10^6|U^H|$.

For any desired variance σ^2, we denote the proof of Gaussian distribution by

$$\textbf{ZKPNormal}(\mathbf{c_x}, \sigma^2)\Big\{(x,r) : \mathbf{c_x} = g^x h^r \wedge x \sim \mathcal{N}(0,\sigma^2)\Big\}.$$

We now briefly analyze its complexity. Recall that given parameter $M = 1/B$, generating a uniformly distributed random number y' has a cost of $O(\log_2(1/B))$ computations and a proof size of $O(\log_2(\log_2(1/B)))$ as described in **ZKPUniform**. The dominant cost of the computation is in the computation of the erf or erfc series. We neglect lower order costs. The number of terms to evaluate is a constant $L = O(\log_2(1/B))$ that the community computes once in advance. Computing

Algorithm 2 Verification phase computations of GOPA

Input: All users have jointly generated a commitment scheme with parameters Θ and each user u has published to the bulletin board:
- $\mathbf{c_{X_u}} = Com_\Theta(X_u, r_{X_u})$ (before the execution of Algorithm 1)
- $\mathbf{c_{\Delta_{u,v}}} = Com_\Theta(\Delta_{u,v}, r_{\Delta_{u,v}})$ (when exchanging with $v \in N(u)$ during Algorithm 1)
- $\mathbf{c}_{\eta_u} = Com_\Theta(\eta_u, r_{\eta_u})$ (after generating η_u)
- $\hat{X}_u$ (at the end of Algorithm 1)
- **ZKPLinear** proofs for Properties (5) and (6)
- **ZKPNormal** and **ZKPRange** proofs for Properties (7) and (8)

for all user $u \in U$ **do**
  Verify that the value inside $\mathbf{c_{X_u}}$ is in $[0, 1]$ (if not, add u to the cheaters list)
  Verify that values inside $\mathbf{c_{X_u}}, (\mathbf{c_{\Delta_{u,v}}})_{v \in N(u)}, \mathbf{c}_{\eta_u}$ sum to $\hat{X}_u$ (if not, add u to cheaters list)
  Verify that the value inside $\mathbf{c}_{\eta_u}$ has distribution $\mathcal{N}(0, \eta_u)$ (if not, add u to cheaters list)
  for all user $v \in N(u)$ **do**
    Verify that values inside $\mathbf{c_{\Delta_{u,v}}}$ and $\mathbf{c_{\Delta_{v,u}}}$ sum to 0 (if not, add u and v to cheaters list)

a term requires three multiplications, a division and a range proof. Evaluating a series hence requires $O(\log_2(1/B))$ multiplications, additions and range proofs. While the first two cost $O(1)$, a range proof costs $O(\log_2(2L+1))$ where $2L+1$ is the size of the range. In this case, $L = O(\log_2(1/B))$ so the cost is $O(\log_2(\log_2(1/B)))$. In total, the cost of the ZKP for the correct computation of the series is therefore $O(\log_2(1/B)\log_2(\log_2(1/B)))$. Depending on the value of x one of two series will be evaluated, and a disjunction of two relations proven with **ZKPOr** (which simply doubles the above cost). Regarding the proof size, the computation of erf or erfc (or the proof of its disjunction) has size $O(\log_2(1/B)\log_2(\log_2(\log_2(1/B))))$, as each of the range proofs performed has a size which is logarithmic in the complexity of its computation. The total proof size is therefore of $O(\log_2(1/B)\log_2(\log_2(\log_2(1/B))))$.

B.4 GOPA Verification Protocol

We design our verification protocol based on the ZKPs described in Appendices B.2 and B.3. Concretely, we rely on **ZKPLinear** to verify Properties (5) and (6), **ZKPNormal** to verify Property (7) and **ZKPRange** to verify Property (8). Note that Property (6) involves secrets of two different users u and v. However, this is not a problem as these pairwise noise terms are known by both involved users, so they can use negated randomnesses $r_{\Delta_{u,v}} = -r_{\Delta_{v,u}}$ in their commitments of $\Delta_{u,v}$ and $\Delta_{v,u}$ such that everybody can verify that $Com_\Theta(\Delta_{u,v}, r_{\Delta_{u,v}}) + Com_\Theta(\Delta_{v,u}, r_{\Delta_{v,u}}) = Com_\Theta(0, 0)$.

At the beginning of the protocol, all users jointly generate parameters Θ for a shared Pedersen commitment scheme (see Appendix C). Our verification protocol, described in Algorithm 2, requires users to publish in the bulletin board all commitments and involved ZKPs as soon as the private values involved are generated by the execution of the private averaging phase (Algorithm 1). The verification can then be performed by any party interested in verifying the integrity of the computation. By composition of ZKPs, every user prove the correctness of his/her computations, and thereby Theorem 4. Completeness, soundness and zero knowledge properties for Theorem 4 are guaranteed as they are preserved by the composition of ZKPs.

In the rest of this section, we give additional details regarding multiple executions and user drop out, and conclude with a discussion of how the correctness guarantees of GOPA compare to the centralized scenario.

B.4.1 Consistency over Multiple Executions

While the guarantees of Theorem 4 hold with respect to a single execution of GOPA, commitments to private values can be used when multiple executions of the algorithm are performed over the same or related data. In addition to linear relations, products and range proofs, a wide variety of arithmetic relations can be proven between committed secrets. As an illustration, consider ridge regression in Example 1. Every user u can publish commitments $\mathbf{c_{y_u}} = Com_\Theta(y_u, r)$, $\mathbf{c_{\phi_u^i}} = Com_\Theta(\phi_u^i, r_i)$ for $i \in \{1, \ldots, d\}$ (computed with the appropriately drawn randomness), and additionally commit to $\phi_u^i y_u$ and $\phi_u^i \phi_u^j$, for $i, j \in \{1, \ldots, d\}$. Then, by the use of **ZKPProd**, it can be verified that all

these commitments are computed coherently, i.e, that the commitment of $\phi_u^i y_u$ is the product of secrets committed in $\mathbf{c_{y_u}}$ and $\mathbf{c_{\phi_u^i}}$ for $i \in \{1, \ldots, d\}$, and analogously for the commitment of $\phi_u^i \phi_u^j$ in relation with $\mathbf{c_{\phi_u^i}}$ and $\mathbf{c_{\phi_u^j}}$, for $i, j \in \{1, \ldots d\}$. An efficient ZKP to check arbitrary arithmetic relations, i.e, inner products of vectors of commitments, is provided in[15] and optimized in [18]. Additionally, ZKPs can be composed such that AND and OR composition of the proven statements can be generated (see **ZKPOr**), and consequently any kind of logical monotone formula [27, 63].

B.4.2 Dealing with User Drop Out

As we are in a decentralized setting, the successful execution of the protocol relies on users remaining active during the computation. In the context of a large number of users, the probability of a few of them to drop out in the middle of the execution must be taken into account. Here, we consider that a user drops out of the computation if he/she is off-line for a period which is too long for the community to wait until his/her return. Malicious users may also intentionally drop out to bias the outcome of the computation. Indeed, drop outs affect the outcome of the computation as we rely on the self-canceling noise terms $\{\Delta_u\}_{u \in U}$ for the correctness and accuracy of the computation.

We propose a three-step approach to handling user drop out. First, as a preventive measure, users should exchange pairwise noise with a few more neighbors than strictly needed to satisfy the privacy requirements, in order to have some margin in case a small number of neighbors drop out. Second, as long as there is time, users attempt to repair as much as possible the problems induced by drop outs. Finally, when circumstances require and allow it,[4] we can ignore the remaining problems and proceed with the algorithm, which will then output a slightly perturbed answer.

We now focus on the second of these three steps. Users can react to the drop-out of a neighbor (i.e., a user with whom they already exchanged pairwise noise) in several ways. First, a user u who did not publish $\hat{X}_u$ yet can just remove the corresponding pairwise noise (and exchange noise with another active user instead if needed). Second, a user u who did publish $\hat{X}_u$ already but has still some safety margin because he exchanged pairwise noise with more neighbors than strictly necessary can simply reveal the noise exchanged with the user who dropped out, and subtract it from his published $\hat{X}_u$.

The third case, in which a user u published $\hat{X}_u$ already but cannot afford to adjust it by removing the noise exchanged by the user who dropped out, is the most difficult one, and should therefore be avoided as much as possible by the preventive surplus of noise exchanges described above. If u is lucky, he/she can still exchange additional noise with other users who did not publish their noisy value yet. However, if u cannot trust that a sufficient fraction of these users are honest, the only choice may be that user u drops himself/herself out. Note that this could cause a cascade of drop-outs if other users also do not have sufficient margin, or a significant part of the pairwise noise exchange procedure needs to be repeated. To avoid such problems, in addition to preventively exchanging surplus pairwise noise, it is best to check which users went off-line just before publishing $\hat{X}$, and to have penalties for users who (repeatedly) drop out at the most inconvenient times.

B.4.3 Comparison of Correctness Guarantees with the Centralized Setting

Consider the case of a centralized computation, where a central party is provided with the raw data to compute the model. In general, this central party does not know whether the values it received as input are truthful. However, it knows that all of its computations are performed correctly given the received input data. In this sense, our commitments offer the same guarantees.

In some critical cases where reliable data is needed, the central party could require that the data of users to be certified by a trusted third party. In such scenarios where credentials are required, solutions exist to provide private credentials to users, that have the same zero knowledge logic as our solution and also provide to the central party a certificate from a trusted third party that this data is valid [19, 17].

We can conclude that GOPA is an auditable protocol that, through existing efficient cryptographic primitives, can offer guarantees similar to the automated auditing which is possible for data shared with a central party.

[4] For instance, only a few drop outs have not yet been resolved, there is not much time available, and the corresponding pairwise terms $\Delta_{u,v}$ are known to be not too large (e.g., by the use of range proofs).

Appendix C Randomness for Setup and ZKPUniform

The generation of challenges in **ZKPUniform** cannot be made non-interactive as it provides an opportunity for malicious users to execute many generations and choose a particular sample, which would bias the distribution. Here, we provide a decentralized method to generate $O(|U|)$ challenges with a computational effort of $O(1)$ per user, and only $O(1)$ messages in the network. Additionally, this algorithm can be used to generate the randomness to run the $Setup$ algorithm and initialize the Pedersen commitment scheme shared by all users.

For this purpose, we make use of a cryptographic hash function $H : \{0,1\}^* \to [0, M-1]$ where M is defined as in the interval of **ZKPUniform**. Cryptographic hash functions are easy to evaluate, but their outcome is impossible to predict or to distinguish from random numbers. Practical instances of such functions can be found in HMAC and Sponge-based constructions for random number generators [7, 9].

To describe our procedure, we enumerate users from 1 to $n = |U|$. The procedure goes as follows:

1. Every user u generates a random number $s_u \leftarrow_R \mathbb{Z}_M$, computes a commitment $\mathbf{c_u} \leftarrow Com(s_u)$ for some randomness and publishes $\mathbf{c_u}$ to the bulletin board.
2. When all commitments are published, every user reveals s_u by publishing it such that all users can check that the commitment $\mathbf{c_u}$ was computed correctly.
3. By setting $s'_1 \leftarrow s_1$ and starting from user 2, user $u+1$ queries the bulletin board to get s'_u, computes $s'_{u+1} \leftarrow s_{u+1} + s'_u \mod M$ and publishes s'_{u+1} in the bulletin board.
4. Users set $t_0 \leftarrow s'_n$ as an unbiased seed for H and sequentially, each user u queries the bulletin board for t_{u-1}, computes its challenge $t_u \leftarrow H(t_{u-1})$ and publishes it to the bulletin board.

The "commit-then-reveal" protocol in Steps 1 and 2 is to avoid users from choosing the value s_u depending on the choice of other users, which could bias the final seed s'_n. As it is computed from modular sums, this seed is uniformly distributed over $[0, M-1]$ if at least one user is honest. As our seed is unpredictable and challenges are obtained by application of H, we make sure that the sequence of challenges $t_1, \ldots, t_n$ is also unpredictable. To obtain more randomness, the protocol above can be repeated.

If users have already initialized parameters Θ of the shared Pedersen scheme, the commitment function Com used is the Pedersen function Com_Θ as defined in Appendix B. Otherwise, a commitment function which does not require the generation of common parameters (but is not suitable for other tasks such as the generation of our ZKPs) can be used, see for example [12]. This happens if the above protocol is used to generate the randomness t as input of the $Setup$ function of the shared Pedersen scheme. In this case, we obtain t by taking as many bits of the sequence $t_1, t_2, \ldots$ as needed.

Note that the request for a challenge t_u is blocking for the user u until all users request a challenge. However, every user needs the same amount of random challenges to execute $Setup$ or **ZKPUniform**. Hence, all these challenges can be requested and used at initialization for $Setup$, and in a moment of the computation fixed by all users for **ZKPUniform**. This prevents that users have to wait for a long time after a request, as would happen in a setting where the requests are imbalanced.

Finally, to verify that a user u participated correctly, one needs to check that (1) the commitment $\mathbf{c_u}$ was properly computed from s_u, (2) s'_u was correctly computed from s_{u-1} and s_u, and (3) the challenge t_u was correctly computed from t_{u-1} by the application of H. This can be added to the verification protocol with $O(1)$ extra computations and messages for every random challenge generated and the execution of $Setup$, and does not have a significant impact in the overall complexity.

Appendix D Complexity of GOPA

The complexity of the protocol in terms of computation and communication was summarized in Theorem 5. This section presents a more detailed analysis of this complexity, relying in particular on the complexity of the zero knowledge proofs discussed in Appendix B.

For computation, we take into account regular operations such as floating point sum, product, division, exponentiation, etc., as well as products and exponentiations in the Pedersen multiplicative group $\mathbb{G}$ described in Appendix B, and cryptographic hash function evaluations. Often, the dominant terms in the computations are discrete exponentiations in $\mathbb{G}$, which still remain lightweight in current implementations of Pedersen commitments [39]. For communication, three types of interactions in the network are taken into account: (1) sending a message to another user in the context of a pairwise exchange, (2) publishing a message or (3) querying the bulletin board for a set of publications. We also measure the size of these messages in the number of values that they contain, which in most cases are elements of $\mathbb{Z}_q$ and $\mathbb{G}$.

The costs break down as follows:

- The initialization of GOPA requires enrolling users for the computation and defining parameters of the Pedersen commitment scheme, which has a cost of $O(1)$ per user. The computation and publication of commitments of the vector X of private values, as well as the verification that every one of its components is in the range $[0, 1]$ (i.e. Property (8)) with **ZKPRange** has a computational and communication cost that is logarithmic in the size of the range $[0, 1]$. As the range proof size is small and constant, this cost is dominated by other costs.
- Interacting with a neighbor to generate a noise term, and publishing a commitment of it requires $O(1)$ computations and messages for a user. Performing the **ZKPLinear** proof to verify Property (6) require $O(1)$ messages and computations (as the linear relation is composed of 2 terms) for a user and its verifier. As $|N(u)|$ pairwise noise terms are generated by user u, this has a cost of $O(|N(u)|)$ for communication and computation in total.
- Generating the private Gaussian distribution variable η_u and proving Property (7) is done by performing **ZKPNormal**. Given parameter B, this requires $O(\log_2(1/B))$ computations for the generation of η_u in the private averaging phase, $O(\log_2(1/B)\log_2(\log_2(1/B)))$ computations to generate and verify the ZKP, and $O(1)$ messages of size $O(\log_2(1/B)\log_2(\log_2(\log_2(1/B))))$.
- Finally, the computation and verification of the proof of Property (5) requires the computation of **ZKPLinear** over $|N(u)| + 3$ terms per user, which has a computational cost of $O(|N(u)|)$ both for the prover and the verifier. The communication cost is of $O(1)$ messages of size $O(1)$ as the private values involved in the proof have already been committed for previous proofs.

The overall computational cost of the private averaging phase for a user $u \in U$ is therefore $O(|N(u)| + \log_2(1/B))$ and the cost of proving its computations in the verification phase is $O(|N(u)| + \log_2(1/B)\log_2(\log_2(1/B)))$. The overall communication cost is composed of $O(|N(u)|)$ messages of size $O(1)$ and $O(1)$ messages of size $O(\log_2(1/B)\log_2(\log_2(\log_2(1/B))))$. Additionally, the complexity of verifying the publications of u is the same as the cost of u proving its computations mentioned above. Finally, the communication cost for the verification of u comes from querying the bulletin board for all of u's publications, i.e. the size of all commitments and ZKPs published, which is $O(|N(u)| + \log_2(1/B)\log_2(\log_2(\log_2(1/B))))$.

As discussed at the end of Appendix B.3, a reasonable value for B in practice is $\sigma_\eta/10^6|U^H|$.

Appendix E Further Discussion on the Impact of Finite Precision

In practice, we cannot work with real numbers but only with finite precision approximations. We provide here a brief discussion of the impact of this on the guarantees offered by the protocol. We emphasize that there is already a large body of work which addresses most of the potential problems which could arise because of finite precision. Here are the main points:

1. Finite precision can be an issue for differential privacy in general, see e.g. [4] for a study of the effect of floating point representations. Issues can be overcome with some care, and our additional encryption does not make the problem worse (in fact we can argue that encryption typically uses more bits and in our setting this may help).
2. The issue of finite precision has been studied in cryptography. While some operations such as multiplication can cause additional trouble in the context of homomorphic encryption, in

our work we use a partially homomorphic scheme with only addition. As a result, we can just represent our floating point numbers with as many bits (after the decimal dot) as our variables also have in computer memory.

3. If bandwidth would dictate that we can transfer only at lower precision compared to the internal representation, a conversion will be needed before encryption (e.g. from 10 digits to 5 digits after the decimal dot). Still, (a) rounding has been well described in numerical computing literature, (b) in our case it would not decrease privacy as all our operations are linear (no sharp differences in behavior if a value changes less than machine precision), and (c) we feel this would not gain much bandwidth: indeed, in practical applications with sums over millions of parties (plus some random encryption bits), one would already require more than 7 digits before the decimal dot, in comparison with which the number of less significant bits gained is relatively small.